

# ENCYCLOPEDIA OF ARTIFICIAL INTELLIGENCE

## VOLUME 2

---

Stuart C. Shapiro, *Editor-in-Chief*

David Eckroth, *Managing editor*

George A. Vallasi, *Chernow Editorial Services, Developmental Editor*

Wiley-Interscience Publication

**John Wiley & Sons**

New York / Chichester / Brisbane / Toronto / Singapore



VISIT...

LANZAROTE  
*Caliente*.COM

108132

Copyright © 1987 by John Wiley & Sons, Inc.

All rights reserved. Published simultaneously in Canada.

Reproduction or translation of any part of this work beyond that permitted by Sections 107 or 108 of the 1976 United States Copyright Act without the permission of the copyright owner is unlawful. Requests for permission or further information should be addressed to the Permissions Department, John Wiley & Sons, Inc.

**Library of Congress Cataloging in Publication Data:**

Encyclopedia of artificial intelligence.

"A Wiley-Interscience publication."

1. Artificial intelligence—Dictionaries.

I. Shapiro, Stuart Charles. II. Eckroth, David.

Q335.E53 1987 006.3'03'21 86-26739

ISBN 0-471-80748-6 (set)

ISBN 0-471-62973-1 (Vol. 2)

**Printed in the United States of America**

10 9 8 7 6 5 4 3

Over-  
size  
Q  
335  
E53  
1987  
v.2

# ENCYCLOPEDIA OF ARTIFICIAL INTELLIGENCE

VOLUME 2

---

**OBJECT-ORIENTED PROGRAMMING.** See Languages, object-oriented.

## OFFICE AUTOMATION

Office work is of two distinct sorts: subject and overhead. The subject work addresses the business of the organization running the office: for a doctor's office, this includes medicine (e.g., diagnosis, treatment) and scheduling (maintaining an appointment calendar); for a sales office, order maintenance, delivery scheduling, compensation, and order forecasting; for an insurance company, financial planning; and for an oil company, geology. In contrast, the overhead is that work done in the process of "running" the office: this concerns people (hiring, firing, paying, directing), physical arrangements (space, desks, chairs), communication (letters, phones, meetings), bureaucracy (forms, policies, procedures), and organization (managing).

### Subject Domains

The domains of subject work cover all of the world's business, a detailed discussion of which is beyond the scope of this article. However, some of these domains are associated more directly with offices, because they appear primarily in office contexts. For example, expert systems (qv) for financial planning, insurance underwriting, legal document preparation, inventory control and sales territory management are all domains traditionally carried out in offices in which AI systems are being used.

Such "expert systems" are most attractive in areas where experts are few, but many are needed (financial advising, insurance underwriting, machinery diagnosis, exploratory geology). Here the expense of an expert system can be justified as enabling less expert people to do the job. Some of the current excitement over AI comes from its notable successes in a number of these domains, particularly ones with high payoff for better answers.

Recently, systems have become available for both mainframes and, more interesting, for personal computers (1), which deliver AI-program construction techniques to nontechnical "users." These provide environments for creating rule sets which potentially capture a domain of interest, and an interpreter for running that rule set on descriptions of particular cases. Sometimes, these construction environments also provide tools for creating attractive user interfaces to the expert systems thus constructed. As a result, AI is playing an increasing role in office systems by acting as the technological base for their construction, making them faster to build and modify and accessible by those lacking the command of more traditional programming practice.

AI programs are not unique here. The very characteristics which make AI-based programs attractive in offices are those which have made word-processing, spreadsheets, and databases attractive: they can be quickly changed and the functionality of the system can be left in the hands of the interested user. In fact, this responsiveness often makes the

resulting systems very responsive to the needs of their (often very specific) domains. The abstractions of those domains appear (unusually uninterpreted) in the systems. Therefore, to an observer, they often appear to understand their domains very well. These similarities add fuel to the debates concerning the definition of AI.

### Routine and Nonroutine Activity

The fundamental fact about offices is that they are inherently composed of people. People make up an office because there is a changing environment within which the parent organization must function. To do this, the people of the organization must constantly be "making sense" of that world, interpreting themselves and the organization in that context.

Since machines cannot sense that world, they cannot interpret it. To the extent that an interpretation can be settled upon, and conveyed to the machine, and that interpretation made to apply for many cases, the machine can be used to handle the growing volume of transactions which make up modern business. To support this, it is necessary that people preprocess the varied world into input expressed in the terms of the chosen interpretation and reinsert through interpretation the machine's output back into that world. This leads to segregating office work into the routine (for machines) and the nonroutine (for people). Thus "office automation" only automates the office in the sense that it transforms the view of office work to enable automation (computers) to play a role.

From this viewpoint, the role of AI-based programs in offices is not significantly different from that of more traditional office computer systems. The AI-based programs being used in offices, while often improvements over less flexible traditional programs, are still incapable of being really responsive to the world in which they are embedded. On the whole technologies are unaware even that there is a situation to be made sense of. Beyond that, they are not capable of conceiving that the characteristics which they address need to be augmented or modified, much less of determining, from confrontation with the actualities of a presented situation, either the nature or the details of those changes. Thus the systems do not work well when used beyond the limits of their expertise. Indeed, they do not as yet have much sense of their own limits (the judgment of which even people find difficult).

Thus in the end, people must interpret these systems to the world. Therefore, the impact of AI is, at most, to change the boundary between the routine and the nonroutine and make that boundary more malleable, but it does not remove it: the essential office, in which people address an ever-changing world, remains unchanged.

### Running the Office: Overhead Domains.

Consider now the overhead of the office, that work which is quintessentially "office work." The office is a sociotechnical environment made up of many systems interacting: physical, technical, intellectual, social, bureaucratic, organizational. The task of an office worker is to do whatever is necessary to achieve certain ends within the organization, with those ends

involving other people, taking place in certain physical locations, with certain technical assists. The organization's goals concern business, but also include legality (2), and responsibility (3,4). Many of these goals are unarticulated. Office work is always changing. The mechanisms and agreements set up by people to get the job done are highly individual and specific. They must be easy to set up and dismantle, often being active for just minutes ("Answer the phone while I make a copy, will you?"). Office work often has a large social component. As an important example, consider the role of a secretary: to catalyze and support communication for the people in the office with each other, the organization and the world.

The account just given of the separation into routine and nonroutine is particularly applicable to office overhead work. Further, application of computers to this domain has been made more difficult by unrealistic views of this work, views which see it as much more routine than it is. The perception of regularity in these views is achieved by looking too narrowly at what is happening. Two good examples are in the domains of flow of paper work and calendars.

Bureaucracy presents the picture of an office as a place where forms flow from worker to worker in a pattern specified by "office procedures." A number of research efforts accepted this apparent regularity and attempted to build systems which would aid in the process, either by modelling it for purpose of analysis (5,6), or by supporting it for purpose of reducing the load on workers in remembering and tracking the flow of "paper" (6,7,8). These efforts were not highly successful because they are based on invalid assumptions about the flow of paper in an office: The set of procedures specifying flow is only one of the factors determining the movement of paper in the office, and the others were not addressed by the systems. Moreover the procedures themselves are only a surface manifestation of a set of underlying goals which the paper flow, indeed the paper itself, is a device for achieving (9).

People in offices use calendars to plan their time. Calendar systems have been built to help people with this task. They have recognized that an important part of this is to help with the coordination required for people to arrange meeting times. These systems have not been generally accepted because they look at the use of calendars too narrowly. First, people move about, and an immobile calendar is simply not acceptable (10). More interesting, because a calendar represents a person's commitments, it also represents priorities, and this is highly personal and contextually sensitive information. Consequently the coordination of meeting times is not a matter of simply finding corresponding available slots in various calendars, but rather is a matter of negotiating, which often turns on complex social and organizational power relationships among those concerned.

All this is not to say that computers and AI will not address these and other domains comprising office overhead work. Indeed, forms and calendar systems have already played an important part in supporting the work done in offices. However, to achieve this, they will have to be based on realistic views of office work. The niceties of the various interactions, particularly the social interactions, are well out of reach of today's AI technology. And although more easily changed than traditional programs, they still cannot change as fast as the office does. Therefore even AI computer systems are best viewed as being in that part of the office which handles the routine part of the work.

## Office Educator: A New Role for AI Systems

One advantage of AI technology over traditional computer systems is that the knowledge about its application domains is made explicit. Thus AI enables a major new role for computer technology within offices: that of educator and communicator. Through AI applied as an adjunct to other systems, both technological and human, people in offices can better come to understand these complex multistructured environments. Through understanding they can gain better command of them. In addition, through this improved command, AI can even enable new environments which are simply not possible without it. More generally, AI can enable changed relationships between office workers and their environments.

It should be noted that this new role for AI is not confined to the office. Explicit knowledge will support the educator's role wherever AI is used. However, because offices are a very common workplace in today's world, AI's role as educator may be more visible to a large nontechnical audience in the office than elsewhere.

The rest of this article discusses four aspects of the office environment concerning which AI may educate: technology, communications, information domains, and the organization itself. This framework provides a perspective on the AI work that has already been done in office systems. Each section indicates how AI systems can provide people with better understanding and command of their environments.

## Understanding Machines

Office machines are becoming more complex. Text editors are difficult to learn (11); manuals for office systems are imposing, even copiers are the subject of study (12). The utility of these machines is limited less by what their developers can provide than by what their users can understand.

A primary role for AI in the offices, then, is to enable better user interfaces. Training tutorials (13,14) can introduce people formally to the machine. Situated online help (15) and intelligent interfaces can assist in the actual use. The studies of what it is to interact with a machine (12) are showing the need for the machine to be able to consider itself as a player in the interaction, and to use its understanding of the interaction itself as a basis for reasoning about that interaction. The work already done on reflection and introspection, reasoning, and repair (16) will be important in achieving this.

One of the most difficult things about the new machines is that they are software-based. The culturally provided way of understanding machinery (as the interaction of comprising subparts) provides no access to this technology. Simulations of the machine while it is running can provide subtle (even enjoyable) support for mental models of the machine which will support reasoning about it (17).

New paradigms for the control of the interaction between user and computer system are also enabled by AI. For example, the work on ONCOCIN (see Medical advice systems) includes exploration of partitioning the responsibility: the user (in this case a doctor treating patients with drugs as part of a medical experiment) is responsible for the dosages given to the patient; the machine is responsible for the acceptability of the dosages given for use in research on treatment methods. To achieve these new styles of interaction between user and machine, the machine must educate the user in the role that they are to play with respect to the machine.

Learning is often best done in a hands-on, "learn through doing" fashion. To support this, the environment must encourage exploration, most particularly by rendering errors hard to make inadvertently or painless to recover from. AI technology has led the way in showing how to do this (18). This attitude will be required by all users, since "closed" systems which can be completely learned will become less and less the rule. A complementary need is to develop in users an attitude that admits partial, incomplete, possible even wrong, understanding of the system as an acceptable state of knowledge: the "purposes at hand" determine what is needed. Online help systems carry this message; dynamically modifiable ones would further it.

Finally, that machines must suit many different styles of usage will require that they be "tailorable" to individual needs. This tailoring can be done explicitly (through specification) or implicitly (e.g., by example). AI can help with each. Explicit specification requires models of the machine and how specification can change it. Tailoring by example requires pattern generalization and recognition (19).

### Understanding Communications

Communications are central to running an office. Indeed communications are the primary subject domain for people like secretaries. AI is beginning to play a role in helping with electronic mail, through aid with addressing (20), with selecting recipients (21), and with sorting mail when received. Voice mail and integrated telephone systems are being explored.

Although these roles for AI will be important, the more profound effects of AI will be in helping people understand the office communication patterns in which they are involved. To large measure, people's knowledge of communications systems is tacit. AI can educate directly by making this information explicit. This should include not only details of the technology (e.g., the need to use protection mechanisms, and the ability to find the person who can supervise them) but also the social mores associated with using it (22) (e.g., who may use a distribution list for what purpose; and expectations, such as how soon messages must be answered, and the social implications of not meeting these expectations). AI can provide descriptions of the communications media, both inside and outside the machine, which will permit users not only to answer the questions concerning the usage of the media, but also to know what questions to ask. Further, with the availability of this information to the communications systems themselves, AI can support the creating of agents for dealing with situated problems (e.g., knowledge-based descriptions of standing-orders) (23).

Educating indirectly, AI can change the very nature of the communications media, providing new opportunities in the office. For example, electronic mail is a new medium with a new set of social values. In particular, it is more formal than the telephone and less formal than office memos. In creating messages, the impact of these felicity conditions on formality is subtle. Having spelling and grammar checkers available may reduce fears associated with sending a message in haste. On the other hand, it may be important to also provide mechanisms which guarantee that a message looks hurried, thereby communicating to the receiver the intent of the sender. Although powerful and potentially very useful, the full impact of such AI systems on communications in the office must be studied with care.

### Understanding Information Domains

One of the cornerstones of AI technology is the explicit representation of the ontology of its application domains. (This is the "knowledge base" of the domain, in terms of which the "data" of the domain is understood and encoded.) In addition to supporting the application, this explicit encoding of domain ontology can be used to teach the user about the domain. This is important in training new personnel and in reducing memory load for experienced users. The presentation can be direct (tools which present the information explicitly to the user) or indirect (e.g., during data acquisition or information retrieval (24)). Explicit ontology can also be used to support its own change. This is particularly useful in the office environment where the ontology of information is often changing rapidly, not only by extension, but also by reinterpretation. With the support of change and the capacity for querying the ontology of the data comes a potential for information change not currently available: the information system can itself act as a communication system for discussing and controlling the terms in which the domains addressed are described (25). This is only one form of "idea generation" in which the systems themselves support the exploration of not only ideas but the frameworks in which those ideas are expressed.

Another dimension of difference between AI ontologies and traditional ones (e.g., data dictionaries, relational databases) is the sophistication of AI knowledge representation (qv) techniques in representing partial, hypothetical, alternative, dependent, and contingent information. This sophistication offers fundamentally new capabilities in the office, where such incomplete and complex information is the rule rather than the exception. AI ontologies also handle heterogeneity in a way not found in traditional ontologies, permitting the various relationships between terms in the ontology to be expressed in detail. Such heterogeneity presents a new solution to an old problem, that of determining, when retrieving information, the terms in which to state that query. Retrieval by reformulation (24) offers a mode of relationship with the information system which contrasts sharply with existing technology: the information system can help with the formulation of the query itself. The user's attitude toward this task can therefore change radically.

Finally, a wholly new continuity to information can be achieved through the explicit representation of the ontology of that information when the changes in that ontology are recorded as well. All the information, both past and present, can be viewed as part of a single structure and the changes in its meaning explored. Tracking the history of information (26) can provide a view of the office which carries with it not only the potential for reviewing that history, but also the possibility for reasoning about it.

### Understanding Organizations

The people in an office are located in a matrix of different structures: physical, technical, intellectual, social, bureaucratic, organizational. When AI capabilities are coupled with communications, the people using the systems can be supplied with information concerning these structures.

As discussed earlier, support of office work has focussed on a bureaucratic view which sees office work as policies, procedures, forms and forms-flow (7,27,28,29). Although this limited view does not have the power to even adequately describe

the work involved in the paperwork of the office, this information is an important driver of that work (9). Making this information explicit within the system offers for the organization all the advantages described above for information spaces: learning, increased sophistication, and history.

More radically, explicit encoding of the structures composing the office matrix offers the possibility of new structures themselves, and hence significantly altered views of the office and office work. At the low end, the explicit encoding of the content of a group discussion can provide new forms of interaction within meetings. "Working at home" can be enhanced by explicit encoding of project information. Distributed work groups may not have either an "office" or a corresponding organizational structure, but rather only agreements to collaborate and a sense of presence of participants maintained through explicitly encoded group information. An even less formal, yet still explicit, form of organization is that provided by distribution lists and teleconferencing within electronic mail. Coupling this with explicit encoding of the structural matrix provides for yet other new interactions within the "office"; the organizationally constructed structure can itself be an important driver of the interactions (21). Finally, small organizations have a fluid structural matrix which makes them particularly productive: there is no need to create organizational or bureaucratic entities to correspond to projects. It may be possible to extend such fluid and complex structures to larger organizations through explicit encoding and manipulation of these organizational relationships.

So by explicitly encoding the structural matrix of an organization and by making that structure a domain for attention, AI technology can be used proactively to create awareness of, and changes in, the processes by which offices and organizations run.

### Summary

AI technology is being used to create office systems for addressing both subject and overhead domains of office work. This technology provides better system construction and maintenance techniques and extended system functionality. However, the essential core of office work is matching the organization to its diverse and changing environment. Since even AI systems are, as yet, not directly responsive to their environments, all office systems must still be regarded as doing the routine work for their human partners who handle diversity and change. But, unlike traditional systems, AI systems explicitly encode information about their domains. Therefore they can extend their limited roles to include that of office educator, providing people in offices with new ways to gain and manipulate the information necessary to command their machines, their communications media, their subject domains and even the organization itself.

### BIBLIOGRAPHY

1. T. J. Schwartz, "Artificial Intelligence in the Personal Computer Environment, Today and Tomorrow," *Proc. of the Ninth Int. Jt. Conf. on Artif. Intell.*, Los Angeles, CA, 1985, 1261-1262.
2. M. A. Boden, "Panel: Artificial Intelligence and Legal Responsibility," *Proc. of the Ninth Int. Jt. Conf. on Artif. Intell.*, Los Angeles, CA, 1985, 1267-1268.
3. H. Thompson, "Empowering Automatic Decision-Making Systems: General Intelligence, Responsibility and Moral Sensibility," *Proc. of the Ninth Int. Jt. Conf. on Artif. Intell.*, Los Angeles, CA, 1985, 1281-1283.
4. Y. Wilks, "Responsible Computers?," *Proc. of the Ninth Int. Jt. Conf. on Artif. Intell.*, Los Angeles, CA, 1985, 1279-1280.
5. G. Bracchi and B. Pernici, "The Design Requirements of Office Systems," *ACM Transaction on Office Information Systems* 2(2), 151-170 (April 1984).
6. G. J. Nutt and C. A. Ellis, "Computer Science and Office Information Systems," SSL-79-6, Xerox Corporation, Palo Alto, CA, 1980.
7. G. Barber, "Supporting Organizational Problem Solving with a Work Station," *ACM Transaction on Office Information Systems* 1(1), 45-67 (January 1983).
8. S. B. Yao, A. R. Hevner, Z. Hi, and D. Luo, "FORMANAGER: An Office Management System," *ACM Transaction on Office Information Systems* 2(3), 235-262 (July 1984).
9. L. A. Suchman, "Office Procedure as Practical Action: Models of Work and System Design," *ACM Transaction on Office Information Systems* 1(4), 320-328 (October 1983).
10. C. M. Kincaid, P. B. DuPont, and A. R. Kaye, "Electronic Calendars in the Office: An Assessment of User Needs and Current Technology," *ACM Transaction on Office Information Systems* 3(1), 89-102 (January 1985).
11. R. L. Mack, C. H. Lewis, and J. M. Carroll, "Learning to Use Word Processors: Problems and Prospects," *ACM Transaction on Office Information Systems* 1(3), 254-271 (July 1983).
12. L. A. Suchman, "Plans and Situated Actions: The problem of human-machine communication," Dissertation. University of California, Berkeley, 1984.
13. J. S. Brown, R. R. Burton, and J. deKleer, "Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III," in D. Sleeman, J. S. Brown, *Intelligent Tutoring Systems* Academic Press, London, 1982.
14. R. R. Burton and J. S. Brown, "An investigation of computer coaching for informal learning activities," in D. Sleeman, J. S. Brown, *Intelligent Tutoring Systems*, Academic Press, London, 1982.
15. G. Fischer, A. Lemke, and T. Schwab, "Knowledge-based Help Systems," *Proc. CHI'85 Human Factors in Computer Systems*, ACM, New York, 161-167, 1985.
16. J. S. Brown and K. VanLehn, "Repair Theory: A Generative Theory of Bugs in Procedural Skills," *Cognitive Science* 4(4), 379-476 1980.
17. T. Moran, "Getting Into a System: External-Internal Task Mapping Analysis," *Proc. CHI'83 Human Factors in Computer Systems*, ACM, New York, 45-49, 1983.
18. W. Teitelman, "Toward a Programming Laboratory," *Proc. of the First Int. Jt. Conf. on Artif. Intell.*, Washington, DC, 1969, 1-8a.
19. D. C. Halbert, "Programming by example," Dissertation. University of California, Berkeley, 1984.
20. D. Tsichritzis, "Message Addressing Schemes," *ACM Transaction on Office Information Systems* 2(1), 58-77 (January 1984).
21. T. W. Malone, K. R. Grant, and F. A. Turbak, "The Information Lens: An Intelligent Sharing in Organizations," *Proc. CHI'86: Human Factors in Computing Systems*, April 1986, 1-8.
22. D. K. Brotz, "Message System Mores: Etiquette in Laurel," *ACM Transaction on Office Information Systems* 3(2), 179-192 (April 1983).
23. T. Kaczmarek, W. Mark, and N. Sondheimer, "The Consul/CUE Interface: An Integrated Interactive Environment," *Proc. CHI'83 Human Factors in Computer Systems*, ACM, New York, 98-102, 1983.
24. F. N. Tou, M. D. Williams, R. E. Fikes, D. A. Henderson, T. W. Malone, "RABBIT: An intelligent database assistant," *Proc. of the Second National Conference on Artificial Intelligence*, AAAI, Pittsburgh, PA, 1982, 314-318.

25. D. A. Henderson, Jr., "The Trillium User Interface Design Environment," *Proc. CHI'86: Human Factors in Computing Systems*, April 1986, 221-227.
26. D. Bobrow and I. Goldstein, "An Experimental Description-based Programming Environment: Four Reports," CSL-81-3, Xerox Corporation, Palo Alto, CA, 1981.
27. R. E. Fikes, "Odyssey: a knowledge-based assistant," *Artificial Intelligence Journal* 3(16), 331-362.
28. R. E. Fikes, "On supporting the use of procedures in Office Work", *Proc. of the First National Conference on Artificial Intelligence*, Stanford, CA, 1980, 202-207.
29. R. E. Fikes, "A commitment-based framework for describing informal cooperative work," *Cognitive Science* 6(4), 331-347, 1982.

D. AUSTIN HENDERSON, JR.  
Xerox PARC

## OPS-5

OPS-5 is a domain-independent production system. It is the latest in a series that started with "OPS" (see C. L. Forgy and J. McDermott, OPS, a Domain-Independent Production System, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, pp. 933-939, 1977). The main source is the OPS-5 User's Manual (see C. L. Forgy, OPS-5 User's Manual, Technical Report, CMU-CS-81-135, Carnegie-Mellon University, Pittsburgh, PA, (July, 1981)). The OPS-5 environment is built on top of LISP (qv) (however, there exists one implementation in "BLISS") and contains a working memory and a production memory. One production consists of a left-hand side (LHS) and a right-hand side (RHS) that correspond approximately to the IF-part and THEN-part of a normal programming language. The RHS of a production is executed if its LHS matches the content of the working memory. If several LHSs match, a conflict-resolution strategy is used. If no LHS matches, the "program" terminates. The VAX configuration program R1 is a commercially used product based on OPS-4 [see J. McDermott, "R1: A rule-based configurator of computer systems," *Artif. Intell.* 19(1), 39-88 (September 1982)]. R1 has been followed up by a new version called XCON (qv).

J. GELLER  
SUNY at Buffalo

## OPTICAL FLOW

When an observer moves relative to the environment, the two-dimensional (2-D) image that is projected onto the eye undergoes complex changes. The pattern of movement of features in the image is referred to as the optical flow field (1). Optical flow can be represented by a 2-D field of velocity vectors as shown in Figure 1. In Figure 1a the optical flow is generated by the movement of an observer relative to a stationary environment. The "observer" is a camera mounted on an airplane that is flying over terrain. A single snapshot from a sequence of images is shown with reduced contrast. The black vectors superimposed on the image represent the optical flow, or velocity field. The direction and length of these vectors indicate the direction and speed of movement of features across the image

as the airplane flies along. Optical flow is also generated by the motion of objects in the environment. Figure 1b shows three views of a three-dimensional (3-D) wire-frame object that is rotating about a central vertical axis. Figure 1c shows a snapshot of the object at a particular moment in time, with vectors superimposed that indicate the velocities of individual points on the object.

The analysis of the optical flow can be divided into two parts: The first is the measurement of optical flow from the changing image, and the second is the use of optical flow to recover important properties of the environment. The motion of features in the image is not provided to the visual system directly but must be inferred from the changing pattern of intensity that reaches the eye. Variations in the measured optical flow across the image (also known as motion parallax) can then be used to recover the movement of the observer, the 3-D shape of visible surfaces, and the locations of object boundaries. For example, from a sequence of optical flows such as that shown in Figure 1a, it is possible to recover the motion of the airplane relative to the ground. The variation in speed of movement of points on the wire-frame object of Figure 1c allows the recovery of its 3-D structure from the changing 2-D projection. Sharp changes in the optical flow field indicate the presence of object boundaries in the scene. The measurement and use of optical flow are discussed below.

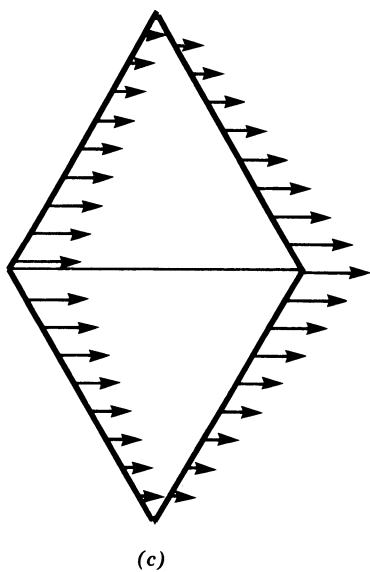
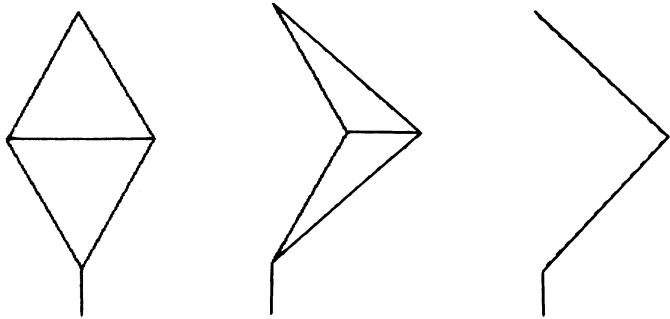
### Measurement of Optical Flow

Computational studies offer a broad range of methods for measuring optical flow (for reviews, see Refs. 2-5). Some methods compute the instantaneous optical-flow field directly. Others track features across the image and thus compute a correspondence between features from one moment to the next. Methods for measuring motion also differ in the stage of image processing at which movement is first analyzed. For example, some infer movement directly from changes in the image intensities, and others first filter the image, or extract features such as edges. The range of techniques for motion measurement (see Motion analysis) are reflected in a broad range of application domains, from the simple tracking of objects along a conveyor belt in an industrial setting to the analysis of more complex motions such as that of clouds in satellite weather data, heart walls in X-Ray images, or cells in cell cultures. The analysis of optical flow is also becoming essential in autonomous navigation (see Autonomous vehicles; Robots, mobile) and robotic assembly (see Computer-integrated manufacturing; Robotics).

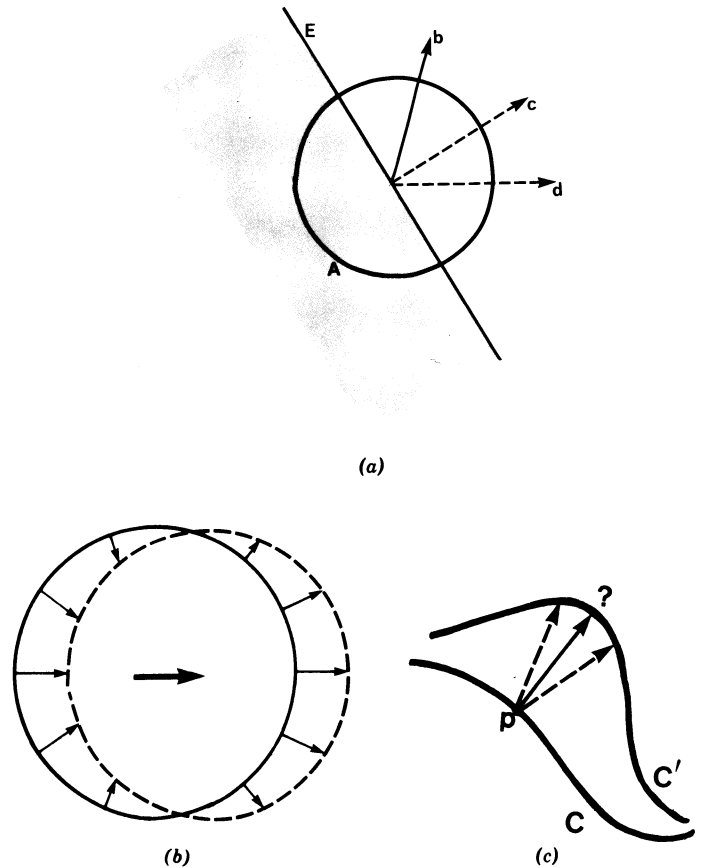
The measurement of optical flow poses two fundamental problems for computer-vision systems. First, the changing pattern of image intensity provides only partial information about the true motion of features in the image due to a problem often referred to as the aperture problem. Second, when the general motion of objects is allowed, there does not exist a unique optical-flow field that is consistent with the changing image. In theory, there exist infinite possible interpretations of the motion of features in the image. Additional constraint is required to identify the most plausible interpretation from a physical viewpoint.

The aperture problem is illustrated in Figure 2. Suppose that the movement of features in the image were first detected using operations that examine only a limited area of the image. Such operations can provide only partial information about the true motion of features in the image (2-7). In Figure





**Figure 1.** (a) Optical-flow field, represented by black arrows, is superimposed on a natural image that was taken from an airplane flying over terrain. (b) Three views of a wire-frame object rotating about a central vertical axis. (c) Projected pattern of velocities of individual points on the object are shown superimposed on a snapshot of the object in motion (an orthographic, or parallel projection is used).



**Figure 2.** (a) Operation that examines the moving edge *E* through the limited aperture *A* can compute only the component of motion *c* in the direction perpendicular to the orientation of the edge. The true motion of the edge is ambiguous. (b) A circle undergoes pure translation to the right. The arrows along the circle represent the perpendicular components of motion that can be measured directly from the changing image. (c) A contour *C* rotates, translates, and deforms to yield the contour *C'*. The motion of the point *p* is ambiguous.

2a the extended edge *E* moves across the image, and its movement is observed through a window defined by the circular aperture *A*. Through this window, it is only possible to observe the movement of the edge in the direction perpendicular to its orientation. The component of motion along the orientation of the edge is invisible through this limited aperture. Thus, it is not possible to distinguish between motions in the directions *b*, *c*, and *d*. This property is true of any motion detection operation that examines only a limited area of the image. As a consequence of the aperture problem, the measurement of optical flow requires two stages of analysis: The first measures components of motion in the direction perpendicular to the orientation of image features; the second combines these components of motion to compute the full 2-D pattern of movement in the image. In Figure 2b a circle undergoes pure translation to the right. The arrows along the contour represent the perpendicular components of velocity that can be measured directly from the changing image. These component measurements each provide some constraint on the possible motion of the circle. Its true motion, however, can be determined only by combining the constraints imposed by these component measurements. The movement of some features such as corners or small patches and spots can be measured unambiguously in the changing image. Several methods for measuring motion

rely on the tracking of such isolated features (2–4,6). In general, however, the first measurements of movement provide only partial information about the true movement of features in the image and must be combined to compute the full optical-flow field.

The measurement of movement is difficult because in theory, there are infinitely many patterns of motion that are consistent with a given changing image. For example, in Figure 2c, the contour C rotates, translates, and deforms to yield the contour C' at some later time. The true motion of the point *p* is ambiguous. Additional constraint is required to identify a single pattern of motion. Many physical assumptions could provide this additional constraint. One possibility is the assumption of pure translation. That is, it is assumed that velocity is constant over small areas of the image. This assumption has been used both in computer-vision studies and in biological models of motion measurement (2–6,8). Methods that assume pure translation are useful for detecting sudden movements and tracking objects across the visual field. These methods have led to fast algorithms for computing a rough estimate of the motion of objects, which is often sufficient in applications of motion analysis. Tasks such as the recovery of 3-D structure from motion require a more detailed measurement of relative motion in the image. The analysis of variations in motion such as those illustrated in Figure 2c requires the use of a more general physical assumption.

Other computational studies have assumed that velocity varies smoothly across the image (5,7). This is motivated by the assumption that physical surfaces are generally smooth. Variations in the structure of a surface are usually small compared with the distance of the surface from the viewer. When surfaces move, nearby points tend to move with similar velocities. There exist discontinuities in movement at object boundaries, but most of the image is the projection of relatively smooth surfaces. Thus, it is assumed that image velocities vary smoothly over most of the visual field. A unique pattern of movement can be obtained by computing a velocity field that is consistent with the changing image and has the least amount of variation possible. The use of the smoothness assumption allows general motion to be analyzed and can be embodied into the optical-flow computation in a way that guarantees a unique solution (5). The optical-flow fields shown in Figure 1 were computed with an algorithm that uses the smoothness assumption (5).

### Use of Optical Flow

Most computational studies of the use of optical flow have focused on the recovery of the movement of the observer and 3-D structure and movement of surfaces on the scene (for reviews; see Refs. 9–11). Some have also addressed the interpretation of discontinuities in the optical-flow field (e.g., Ref. 12). Theoretically, the two problems of recovering the 3-D movement of the environment and observer are closely related. A fundamental problem faced by both is that there does not exist a unique interpretation of the 2-D image in terms of the 3-D motion of the observer and visible surfaces. Additional constraint is required to obtain a unique interpretation.

With regard to the recovery of 3-D structure from motion, computational studies have used the assumption of rigidity to derive a unique interpretation. These studies assume that if it is possible to interpret a changing 2-D image as the projection of a rigid 3-D object in motion, such an interpretation should

be chosen (for reviews, see Refs. 4,9–11). When the rigidity assumption is used in this way, the recovery of structure from motion requires the computation of the rigid 3-D objects that would project onto a given 2-D image. The rigidity assumption was suggested by perceptual studies that described a tendency for the human visual system to choose a rigid interpretation of moving elements (13,14).

Computational studies have shown that it is possible to use the rigidity assumption to derive a unique 3-D structure from a changing 2-D image. Furthermore, it is possible to derive this unique 3-D interpretation by integrating image information only over a limited extent in space and in time. For example, suppose that a rigid object in motion is projected onto the image using a parallel projection such as that illustrated in Figure 1c. Three distinct views of four points on the moving object are sufficient to compute a unique rigid 3-D structure for the points (9). In general, if only two views of the moving points are considered or fewer points are observed, there are multiple rigid 3-D structures consistent with the changing 2-D projection. If a perspective projection of objects onto the image is used instead, two distinct views of seven points in motion are usually sufficient to compute a unique 3-D structure for the points (11). Other theoretical results address the information that can be obtained directly from the optical-flow field (8,9). These and other theoretical results are summarized in Ref. 9. These results are important for two reasons. First, they show that by using the rigidity assumption, it is possible to recover a unique structure from motion information alone. Second, they show that it is possible to recover this structure by integrating image information over a small extent in space and time. Computational studies of the recovery of structure from motion provide algorithms for deriving the structure of moving objects (9–11). These algorithms can also be used to recover the motion of an observer relative to a stationary scene.

### BIBLIOGRAPHY

1. J. J. Gibson, *The Perception of the Visual World*, Houghton Mifflin, Boston, MA, 1950.
2. W. B. Thompson and S. T. Barnard, "Low-level estimation and interpretation of visual motion," *IEEE Comput.* **14**, 47–56 (1980).
3. S. Ullman, "Analysis of visual motion by biological and computer vision systems," *IEEE Comput.* **14**, 57–69 (1981).
4. D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
5. E. C. Hildreth, *The Measurement of Visual Motion*, MIT Press, Cambridge, MA, 1984.
6. D. T. Lawton, "Processing translational motion sequences," *Comput. Vis. Graph. Img. Proc.* **22**, 116–144 (1983).
7. B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.* **17**, 185–203 (1981).
8. K. Nakayama, "Biological image motion processing: A review," *Vis. Res.* **25**, 625–660 (1985).
9. S. Ullman, Recent Computational Studies in the Interpretation of Structure and Motion, in J. Beck, B. Hope, and A. Rosenfeld (eds.), *Human and Machine Vision*, Academic, New York, 1983, pp. 459–480.
10. H. C. Longuet-Higgins and K. Prazdny, The Interpretation of a Moving Retinal Image in S. Ullman and W. Richards (eds.), *Image Understanding 1984*, Ablex, Norwood, NJ, 1984, pp. 179–193.
11. R. Y. Tsai and T. S. Huang, Uniqueness and Estimation of 3-D

- Motion Parameters and Surface Structures of Rigid Objects. in S. Ullman and W. Richards (eds.), *Image Understanding 1984*, Ablex, Norwood, NJ, 1984, pp. 135–171.
12. K. M. Mutch and W. B. Thompson, "Analysis of accretion and deletion at boundaries in dynamic scenes," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-7**, 133–138 (1985).
  13. H. Wallach and D. N. O'Connell, "The kinetic depth effect," *J. Exper. Psychol.* **45**, 205–217 (1953).
  14. G. Johansson, "Visual motion perception," *Sci. Am.* **232**, 76–88 (1975).

E. HILDRETH  
MIT

This document was produced using the facilities of the MIT Artificial Intelligence Laboratory, supported in part by DARPA contract N00014-80-C-0505.

# P

## PAM

A goal-based story-understanding system (see Story analysis), PAM was written in 1978 by Wilensky at Yale University (see R. Wilensky, PAM, in R. C. Schank and C. K. Riesbeck (eds.), *Inside Computer Understanding: Five Programs Plus Miniatures*, Lawrence Erlbaum, Hillsdale, NJ, pp. 136–179; 1981).

M. TAIE  
SUNY at Buffalo

## PANDEMONIUM

Developed in 1959 by O. G. Selfridge, PANDEMONIUM was one of the earliest attempts at machine learning (qv). It exemplified the approach to learning at the time, anthropomorphic neural networks with partially random initial structure. Research on learning has moved away from this paradigm (see O. G. Selfridge, Pandemonium: A Paradigm of Learning, in D. Blake and A. Uttley (eds.), *Proceedings of the Symposium on Mechanization of Thought Processes*, Her Majesty's Stationery Office, London, 1959).

K. S. ARORA  
SUNY at Buffalo

## PARRY

PARRY was one of the earliest attempts at belief modeling (see Belief systems). Designed by Colby around 1971 at Stanford University, PARRY simulated the conversational behavior of a paranoid person. The system integrates inferences with affects and intentions to produce behavior that has been classified as paranoid by several psychologists who conversed with the program (see K. Colby, *Artificial Paranoia*, Pergamon, New York, 1975).

K. S. ARORA  
SUNY at Buffalo

## PARSING

Parsing a sentence of a natural language such as English is the process of determining if it is syntactically well formed (grammatical) and, if so, of finding one or more structures (structural descriptions) that encode useful information of some kind about it. The word is derived from the Latin *pars orationis* (part of speech) and reflects a process that has been carried out by human beings from medieval times to the present. This activity traditionally took the form of assigning a part of speech to every word in a given sentence, of determining the grammatical categories of words and phrases, and of enumerating the grammatical relations between words. Its purpose was pedagogical, to help students of a language increase their mastery of it.

In modern times developments in linguistics and computer science have led to a somewhat different set of activities being associated with the term *parsing*. The availability of computers was one of the factors that led to the replacement of vague, partially specified procedures that were carried out by humans by well-specified algorithms that were carried out by machines. Also, the change in purpose of the activity led to a corresponding change in the nature of the structural descriptions produced. Pedagogical concerns were replaced by a requirement that structural descriptions reflect the meaning(s) of the sentences. This is of special importance in AI applications, in which the intent of input sentences must be understood and acted upon in an appropriate manner.

Still another change stemmed from the use of formal systems to model aspects of natural languages. In particular, many different types of formal, generative grammars have been devised to specify the sentences of a language and to pair each sentence with a corresponding set of structural descriptions. The nature of these grammars, however, usually does not provide an obvious algorithm for computing structural descriptions from sentences. In this respect they are similar to systems of logic, which implicitly specify a set of provable theorems but do not explicitly tell how a particular theorem is to be proved. Just as proof procedures must be devised for systems of logic, so must parsing procedures be devised for formal grammars of natural languages. Parsing a given sentence

with respect to a given grammar, then, is the process of determining whether the sentence belongs to the language specified by the grammar and, if so, finding all the structures that the grammar pairs with the sentence.

The most common type of grammar used within computer science to syntactically specify the sentences of a particular programming language and to assign structure to them is the BNF (Backus–Naur form) grammar. This is a notational variant of a class of grammars called *context-free* (CF) grammars, which play a prominent role in many computational and AI models of natural language. It is worth remarking, however, that there is a central difference between the use of CF grammars in computer science and in computational linguistics. In the former, subclasses of CF grammars are utilized that are both unambiguous (i.e., they assign at most one structural description to a sentence) and parsable in time linearly proportional to the length of the sentence parsed. In the latter, however, use is made of a parsing algorithm either for the general class of (ambiguous) CF grammars or for an even more general class of grammars, which often makes some use of CF grammars. For this reason we shall treat in some detail the parsing of CF grammars. Readers who are familiar with the BNF specification of programming languages may wish to skip the next section, which contains introductory material on phrase-structure grammars. The subsequent sections return to the central problem of parsing.

### Phrase-Structure Grammars

The four components of a phrase-structure grammar are a set of symbols from a terminal vocabulary  $V_T$  (terminals), another disjoint set of symbols from a nonterminal vocabulary  $V_N$  (nonterminals), a distinguished element of  $V_N$  called the start symbol, and a set of rules or productions  $P$ . By suitably replacing restrictions on the allowable productions, different types of phrase-structure grammars may be specified. The previously mentioned CF grammar is one such type. All of its rules are of the form  $A \rightarrow A_1 A_2 \cdots A_n$  where  $A$  belongs to  $V_N$ , and the  $A_i$  belong either to  $V_T$  or  $V_N$ . The CF right member of a production is the empty string, and such a production is called an erasing rule.

A *derivation* with respect to a CF grammar  $(V_N, V_T, S, P)$  is a sequence of strings, the first of which is the start symbol  $S$ , and each subsequent member (*sentential form*) is producible from its predecessor by replacing one nonterminal symbol  $A$  by a string of terminal and nonterminal symbols  $A_1 A_2 \cdots A_n$  where  $A \rightarrow A_1 A_2 \cdots A_n$  is a production of  $P$ . Sentential forms consisting entirely of terminal symbols can have no successors in a derivation, and the set of all such terminal sentential forms is said to constitute the language specified by the given grammar  $(V_N, V_T, S, P)$ .

By way of illustration, consider the CF grammar  $G_1 = (V_N, V_T, S, P)$  where  $V_N = (S, NP, VP, DET, N, V, PP, PREP)$ ,

$V_T = (I, the, a, man, park, telescope, saw, in, with)$

(Terminals such as “telescope” and nonterminals such as  $PREP$  are to be regarded as atomic symbols), and  $P$  is the set of productions:

$S \rightarrow NP VP$	$N \rightarrow man/park/telescope$
$VP \rightarrow V NP/VP PP$	$DET \rightarrow the/a$
$NP \rightarrow I/NP PP/DET N$	$V \rightarrow saw$
$PP \rightarrow PREP NP$	$PREP \rightarrow in/with$

The BNF abbreviatory convention is used for writing

$VP \rightarrow V NP/VP PP$

to indicate the two productions  $VP \rightarrow V NP$  and  $VP \rightarrow VP PP$ . A sample derivation is the sequence of sentential forms:

$S, NP VP, I VP, I VP PP, I V NP PP, I saw NP PP,$   
 $I saw DET N PP, I saw the N PP,$

$I saw the man PP, I saw the man PREP NP,$

$I saw the man in NP, I saw the man in DET N,$

$I saw the man in the N, I saw the man in the park.$

The final sentential form, “I saw the man in the park,” is one of the sentences in the language generated by  $G_1$ .

$G_1$  has been made simple to aid in illustrating certain parsing algorithms, but it is deficient in generating such sentences as “a park in I saw I.” A much more complicated set of productions is required to produce reasonable coverage of a natural language without generating such unwanted strings of terminals.

Requiring that the structural descriptions reflect meaning makes the task of producing adequate grammars much more difficult. The use of derivations can be extended to provide for structural descriptions by replacing each production  $A \rightarrow A_1 A_2 \cdots A_n$  by a corresponding production  $A \rightarrow ({}_A A_1 A_2 \cdots A_n)$  where  $({}_A$  and  $)$  are two new terminal symbols.

The result of such systematic replacement of productions  $P$  and augmentation of  $V_T$  for  $G_1$  is another CF grammar,  $G_2$ . For every derivation of  $G_1$  there is a derivation of  $G_2$  in which corresponding productions are invoked. The structural description of a sentence generated by  $G_1$  is the sentence generated by the corresponding derivation with respect to  $G_2$ . For the example, that derivation is

$S, ({}_S NP VP), ({}_S ({}_S NP I) VP), \dots,$   
 $({}_S ({}_S NP I) ({}_V ({}_V ({}_V saw) ({}_N ({}_N DET the) ({}_N man))))$   
 $({}_P ({}_P ({}_P PREP in) ({}_N ({}_N DET the) ({}_N park))))$

This last sentential form of the derivation with respect to  $G_2$  is the structural description of “I saw the man in the park.” It is a labeled bracketing that is one notation for expressing the tree structure shown in Figure 1.

This representation is easier for humans to assimilate but takes more space, and henceforth the labeled bracketing format will be used to represent trees, further simplifying it to

$(S (NP I) (VP (VP (V saw) (NP (DET the) (N man))))$   
 $(PP (PREP in) (NP (DET the) (N park))))$

Note that there is a second structural description of “I saw the man in the park”:

$(S (NP I) (VP (V saw) (NP (NP (DET the) (N man))))$   
 $(PP (PREP in) (NP (DET the) (N park))))$

which groups the words in such a way that the string “the man in the park” is a single constituent, an  $NP$ , leading naturally to the interpretation that the man was in the park when he was seen. The first structural description, however, does not

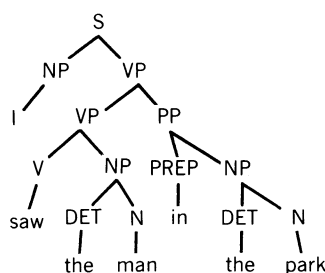


Figure 1.

group the string "the man in the park" as a single constituent. Instead, the VP "saw the man" is a sister to the PP "in the park," indicating the interpretation that the location of the seeing of the man was in the park.

Other types of phrase structures such as finite-state grammars, context-sensitive grammars, and unrestricted rewriting systems (1-3) are definable by placing suitable restrictions on the type of productions allowed. They play a lesser role in specifying natural languages and hence are not treated here.

A topic of some importance in parsing is that of equivalence between two grammars. Grammars that generate the same language are said to be weakly equivalent. The term *strong equivalence* has been used to imply at least the existence of a 1:1 correspondence between the structural descriptions of two grammars, and in some definitions of strong equivalence, the existence of a trivial homeomorphism from the structures of one grammar to those of another has been required. The importance of equivalent grammars to parsing is that there are several circumstances in which it is preferable not to parse a given grammar directly but rather to construct from it an equivalent grammar and to parse with respect to it instead. In certain cases the structural descriptions assigned by the equivalent grammar are just as useful as those assigned by the given grammar. In other cases structural descriptions with respect to the given grammar are needed, but it still may be more efficient to obtain an equivalent grammar, parse with respect to it, and convert the resulting structural descriptions to those of the given grammar than to parse directly with respect to that given grammar. There is another reason for using equivalent grammars in parsing. Sometimes a particular parsing algorithm is only valid for grammars with productions satisfying some restrictions, and it can be shown that for an arbitrary grammar, an equivalent grammar that satisfies those restrictions can be constructed. Often such restrictions are not essential for a particular parsing algorithm but greatly simplify its exposition and hence are useful for pedagogical purposes.

Three types of equivalent grammars for CF grammars much used in parsing are those with no erasing rules, Chomsky normal-form grammars, and Greibach normal-form grammars (1,2). Using capital letters to denote nonterminals and lowercase letters to denote terminals, Chomsky normal-form grammars are those with rules  $A \rightarrow BC$  or  $A \rightarrow a$ , and Greibach normal-form grammars are those with rules  $A \rightarrow aA_1A_2 \dots A_n$  where  $A_1A_2 \dots A_n$  is a string of zero or more nonterminals.

Another topic to be considered in this section relates to the encoding of complex information in the nodes of structural description trees and to generalizing productions so as to require that the nodes they involve have prescribed information. This information usually takes the form of *features* and their

*values*. Feature names are atomic entities such as ANIMATE, NUMBER, and GENDER and feature values are sometimes binary (+ or -), sometimes  $n$ -valued but atomic such as MASCULINE, FEMININE, and NEUTER, and sometimes complex such as a set of recursively used feature-value pairs. Often, the use of such features can enormously simplify the complexity of the set of productions required to specify a particular language. Also, it can be shown that such use of features can be made in ways that do not affect the class of languages definable. Thus, for example, it is possible to extend CF grammars with feature restrictions in such ways that the languages definable are precisely those definable by ordinary CF grammars. This use of features has played an important role in restoring to favor the use of CF languages in natural-language specification and parsing. They are discussed further in a subsequent section.

The final subject to be discussed in this section involves the equivalence between phrase-structure grammars and corresponding automata. All of the types of phrase-structure grammars mentioned have as counterparts corresponding types of automata. Finite-state automata correspond to finite-state grammars, pushdown automata correspond to CF grammars, linear bounded automata to CS grammars, and Turing machines to unrestricted rewriting systems. By correspondence is meant the existence of constructions from grammars to automata and vice versa that preserve the languages specified and the structural descriptions assigned to their sentences.

There are several ways in which this correspondence can be exploited; the one of most concern regards the use of automata to model particular parsing algorithms and to model ways of implementing those algorithms. The space available here allows only informally describing parsing algorithms rather than specifying them precisely by means of automata, but the reader is directed to other sources for instances of such usage of equivalent automata (2,4). In the next three sections, three of the most commonly used CF parsing algorithms are examined: recursive-descent, left-corner, and chart.

### Recursive-Descent Parsing

Recursive-descent parsing, sometimes just called top-to-bottom CF parsing (1,3,4), systematically pieces together structural description trees from top to bottom and from left to right. At each stage of parsing the leftmost unexpanded nonterminal is identified, and its daughter nodes are attached using one of the productions that rewrite that nonterminal. If there is more than one such production, the parser tries them all, following a separate continuation path in each case. Such a process is called *nondeterministic*. Its implementation is often achieved by using a pushdown list to store continuations that are subsequently retrieved and followed.

Terminal symbols thus incorporated into a structural description are matched against the next symbols of the string being parsed. Failure to match causes the continuation in question to fail or block. A continuation also fails if there are remaining input string symbols after the last nonterminal has been expanded.

One successful parse path is given by:

S  
(S NP VP)  
(S (NP I) VP)

(S (NP I) (VP V NP))  
 (S (NP I) (VP (V saw) NP))  
 (S (NP I) (VP (V saw) (NP NP PP)))  
 (S (NP I) (VP (V saw) (NP (NP DET N) PP)))  
 (S (NP I) (VP (V saw) (NP (NP (DET the) N) PP)))  
 (S (NP I) (VP (V saw) (NP (NP (DET the) (N man)) PP)))  
 (S (NP I) (VP (V saw) (NP (NP (DET the) (N man))  
 (PP PREP NP))))  
 (S (NP I) (VP (V saw) (NP (NP (DET the) (N man))  
 (PP (PREP in) NP))))  
 (S (NP I) (VP (V saw) (NP (NP (DET the) (N man))  
 (PP (PREP in) (NP DET N)))))  
 (S (NP I) (VP (V saw) (NP (NP (DET the) (N man))  
 (PP (PREP in) (NP (DET the) (N man)))))  
 (S (NP I) (VP (V saw) (NP (NP (DET the) (N man))  
 (PP (PREP in) (NP (DET the) (N park)))))

Another successful parse path leads to the structural description

(S (NP I) (VP (VP (V saw) (NP (DET the) (N man)))  
 (PP (PREP in) (NP (DET the) (N park)))))

whose structure indicates the sequence of continuations involved in producing it.

Note that whenever the production  $NP \rightarrow NP PP$  is used to expand a leftmost nonterminal, the resulting structure continues to have NP as its leftmost unexpanded nonterminal. Hence, expansion via this rule takes place indefinitely, and it is seen that the algorithm does not terminate. More generally, it is observed that nontermination of the recursive descent algorithm will occur whenever it is applied to a grammar that admits recursive left branching, i.e., a leftmost derivation from some nonterminal A to a string beginning with A.

There are several ways of ensuring termination of this algorithm. First, the grammar to be parsed can be required to disallow recursive left branching. This is not as serious a limitation as might at first be expected because it has been shown that there are constructions that map a given CF grammar into an equivalent CF grammar that is not left recursive. Constructions to Greibach normal form can be used for this purpose. One such construction is due to Rosenkrantz (5). A second way of ensuring termination of the algorithm is applicable to grammars that contain no erasing rules. For such grammars a continuation can be blocked whenever the number of nonterminals that remain to be expanded exceeds the number of still unmatched terminal symbols in the input string.

An obvious improvement that can be made to recursive-descent parsing involves the use of a left-branching reachability matrix R with elements  $R(A,B)$  whose arguments A and B range over the union of  $V_T$  and  $V_N$ .  $R(A,B)$  is T (true) or F (false) depending, if B belongs to  $V_N$ , on whether it is possible to left branch down from B to A, and, if B belongs to  $V_T$ , depending on whether  $A = B$ . That is,  $R(A,B)$  is T if the grammar in question has a derivation from B to a string begin-

ning with A or if  $A = B$ . Note that R depends only on a given CF grammar, not on any particular string to be parsed with respect to it. Thus, R can be computed once and for all for a given grammar of interest. Marshall's algorithm (6) for computing R has been proved optimal and is therefore recommended.

Use of matrix R is illustrated by the following continuation:

S  
 (S NP VP)  
 (S (NP DET N) VP)

The leftmost unexpanded nonterminal DET cannot left branch down to the next input string terminal "I" because  $R(I,DET) = F$ . Hence, this continuation can be blocked at this point without having to consider further continuations that result from expanding DET.

### Left-Corner Parsing

As the name implies, left-corner parsing (called SBT parsing in Ref. 4) builds sentence structures in a left-to-right, bottom-to-top fashion, piecing together the left corner of a structural description tree first. It is not the only parsing algorithm that builds structure from bottom to top. Shift-and-reduce parsing is one of the more commonly encountered cases in point (1,3,4). At each step in left-corner parsing, having determined a left-corner subtree of a structural description tree, it attempts to extend that subtree by scanning the productions for those whose right members begin with the root node of the left-corner subtree. Substituting that subtree for the first constituent of the right member of such a production gives a larger left-corner subtree; all of the daughter nodes of its root node except the first remain to be replaced by appropriate structure, this being accomplished in left-to-right order, recursively using this same left-corner parsing algorithm.

Once again, this algorithm is nondeterministic. There can be more than one production with a right member beginning with a given constituent, leading to one type of nondeterminism. Another source of nondeterminism arises whenever a subtree is successfully built up to replace a constituent other than the first one in the right member of some production. In addition to making the replacement, it is also necessary to attempt to build the subtree up to a larger subtree with the same root node.

As with recursive-descent parsing, the recursive left-branching matrix  $R(A,B)$  can be used to curtail continuations that must eventually fail. At each point where a left-corner subtree has been built up, one knows the nonterminal that one is next attempting to satisfy (i.e., to replace). If that subtree has root A, and B is the nonterminal to be satisfied, then one should block if  $R(A,B) = F$  without attempting further left-corner building of this subtree.

The first few steps in one successful path for the left-corner parsing of the sentence "I saw the man in the park" with respect to the grammar G1 are the following: The only production whose right member begins with the first word in the sentence to be parsed, "I," is  $NP \rightarrow I$ . This gives the left-corner subtree (NP I), and one of the productions whose right member begins with the root of this subtree is  $S \rightarrow NP VP$ . Letting the left-corner subtree satisfy the NP node of this production gives



the new (partially determined) subtree (S (NP I) VP). One must still parse the remaining string "saw the man in the park" up to a tree with root VP to satisfy the VP node in (S (NP I) VP). Proceeding in the same way, the left-corner subtree (V saw) and then the partially determined subtree (VP (V saw) NP) are obtained. The remaining input string at this point is "the man in the park." An initial substring of it must be left-corner-parsed up to a subtree with root NP. Suppressing the details of this, it turns out to be (NP (DET the) (N man)), with "in the park" left as the remaining portion of the input string. This subtree is used to replace the NP node in the previous structure (VP (V saw) NP), giving the new left-corner subtree (VP (V saw) (NP (DET the) (N man))). Next one of the productions is chosen whose right member begins with VP,  $VP \rightarrow VP PP$ , and it is combined with the previously determined left-corner subtree to obtain (VP (VP (V saw) (NP (DET the) (N man))) PP). Finally, the remaining input string "in the park" is left-corner-parsed up to a subtree with root node PP, and this subtree is used to replace the PP in the previous structure. The result is one of the required structural descriptions, (S (NP I) (VP (VP (V saw) (NP (DET the) (N man))) (PP (PREP in) (NP (DET the) (N park))))).

The Rosenkrantz equivalent grammar construction was previously mentioned as a means of eliminating left branching. It is also of value in relating recursive-descent parsing and left-corner parsing. Griffiths and Petrick (7) have shown that the left-corner parsing of a given CF grammar is mimicked by the recursive-descent parsing of the corresponding Rosenkrantz equivalent grammar. This has been exploited in parsing efforts making use of the PROLOG programming language. The productions of a CF grammar can be directly transcribed into a PROLOG form that permits parsing without programming any parsing algorithm. PROLOG itself provides a top-down, depth-first search (qv) procedure, which has the effect of performing recursive-descent parsing (see Recursion). Although PROLOG does permit the easy implementation of a CF parser, its value is limited by the limitations of recursive-descent parsing, namely that it does not allow recursive left branching, and it is slower than such alternatives as left-corner parsing and chart parsing for most grammars of practical interest. To avoid both of these problems, the Rosenkrantz equivalent grammar construction (programmed in PROLOG) has been used to obtain a grammar that can be parsed via the PROLOG recursive-descent procedure, thus mimicking left-corner parsing of the original grammar.

Griffiths and Petrick also used the Rosenkrantz construction to prove that the time required for left-corner parsing is, at worst, a constant (depending on the grammar) multiple of the time required to parse the sentence with respect to the same grammar by recursive descent (8).

### Chart Parsing

All of the parsing algorithms described to this point have worst-case exponential upper bounds. That is, there exist grammars and sentences whose parsing requires a number of steps proportional to a constant raised to the power of the number of words in the input string of words. The reason for this is that when two or more nondeterministic continuations arise, each of them can lead to the identical determination of some substructure common to them all. To avoid this, it is possible to store information as to which subtrees have been

found to span substrings of the input string. This information can then be looked up to avoid computing it more than once.

There are a number of different chart-parsing algorithms. One variant, separately discovered by Cocke, Kasami, and Younger (9), is now usually referred to as CKY parsing. (See Parsing, Chart). Like most of the other chart-parsing algorithms, it has a worst-case upper bound proportional to the cube of the length of the input string. Other chart-parsing algorithms of note are due to Kay (10), Earley (11), Kaplan (12), and Ruzzo (13). The latter is both especially simple and efficient, having a worst-case upper bound proportional to the cube of the input string length with an attractively small constant of proportionality.

Ruzzo's parsing algorithm makes use of a chart or matrix whose elements  $t_{i,j}$  ( $0 \leq i \leq j \leq n$ ) are determined during the course of parsing a string of length  $n$ . Each element  $t_{i,j}$  consists of a set of items each of which is a production of the given grammar with a single dot located somewhere among the constituents of the right member. For example,  $PP \rightarrow PREP DOT NP$  is a typical item. The positions between the terminals of the input string are numbered as in (0 I 1 saw 2 the 3 man 4 in 5 the 6 park 7). If element  $t_{i,j}$  contains the item  $A \rightarrow A_0 \dots A_k DOT A_{k+1} \dots A_m$ , this indicates that the input string between points  $i$  and  $j$  has been parsed up to a string of trees whose roots are  $A_0 A_1 \dots A_k$ , and if some string beginning at point  $j$  can be parsed up to a string of trees with roots  $A_{k+1} \dots A_m$ , the concatenation of both those strings of trees can be parsed up to the parent node  $A$  to obtain a tree with root  $A$ .

For grammar G1 the items of  $t_{0,0}$  are (S  $\rightarrow$  DOT NP VP, NP  $\rightarrow$  DOT I, NP  $\rightarrow$  DOT NP PP, NP  $\rightarrow$  DOT DET N, DET  $\rightarrow$  DOT the, and DET  $\rightarrow$  DOT a. They are obtained by taking the productions that begin with a constituent  $A$  such that  $R(A, S) = T$ , where  $R$  is the left-branching reachability matrix, and forming items in which the dot is located in front of the first constituent. These items indicate the initial possibilities. Three types of actions fill in the elements of the matrix  $t_{i,j}$ . Elements  $t_{i,j}$  are filled in by generating items of the type we have seen for  $t_{0,0}$  in much the same manner that has already been illustrated. Some of the items of elements  $t_{i,j}$  are formed from those of  $t_{i,j-1}$  by hopping the dot one position to the right if the constituent hopped over is either the input string terminal located between points  $j-1$  and  $j$  or else a nonterminal from which there is a derivation to that input terminal. The remaining items are filled in by considering, in the proper sequence, pairs of items, the first of which is of the form  $A \rightarrow A_0 \dots A_p DOT A_{p+1} \dots A_q$  and the second of which is the form  $A_{p+1} \rightarrow \dots DOT$ . If such a pair comes from corresponding elements  $t_{i,j}$  and  $t_{j,k}$ , this indicates that the substring of terminals from points  $i$  to  $j$  has been parsed up to the string  $A_0 \dots A_p$  with a possible continuation of  $A_{p+1}$ , and the substring from points  $j$  to  $k$  has been parsed up to  $A_{p+1}$ , realizing that possibility. Hence, the dot is hopped over the constituent to its right in the first item, and the resulting item is included among the items of  $t_{i,k}$ .

Acceptance is indicated by the presence of an item of the form S  $\rightarrow \dots DOT$  in  $t_{0,n}$ . Structural descriptions are easily obtained by modifying the form of items described above to indicate the tree structure of the constituents to the left of the dot. Whenever the dot is hopped over a constituent, that constituent is replaced by any structure it dominates. The necessary information about this structure comes either from the other item involved or from information supplied by the gram-

mar about a derivation of the next terminal symbol from the hopped-over nonterminal.

### Complex Feature-Based Grammars

There are a number of types of grammars of current interest for specifying natural languages that make some central use of complex feature-augmented rules and structures. To various degrees, they also incorporate other formal devices in addition to their central phrase-structure components. They share a common requirement for matching, in a certain way, rules containing complex features with structural description trees or tree fragments containing complex features, and hence they are sometimes referred to as unification-based grammars. Examples of such grammars are generalized phrase-structure grammars (see Grammar, generalized-phrase-structure) (14), definite clause grammars (see Grammar, definite-clause) (15), functional unification grammars (16), head grammars and head-driven grammars (17), lexical functional grammars (18), and PATR-II-type grammars (19).

It is beyond the scope of this entry to describe these formalisms sufficiently to describe their parsers. Note that they all incorporate the use of a suitably-modified phrase-structure grammar parser (see Grammar, phrase-structure), usually one of the more common types of parsers. See the sources cited for detailed information about these grammars and their parsers.

Two other types of grammars might also be included among those of this section because their rules and structures also make essential use of complex features. These two, transformational grammars and augmented transition network grammars, however, are treated separately in the subsequent sections.

### Transformational Grammar Parsing

It should first be noted that there is no single theory that is agreed upon by all who use the term *transformational grammar* (TG) to label the syntactic theory they advocate (see Grammar, transformational). This is so because TG has evolved, splitting on occasion into distinct formal models of language with significant differences in the type of rules that are allowed, on the constraints imposed on rule application, and on the type of structural descriptions and intermediate structures that are advocated.

It is beyond the scope of this entry to discuss these diverse types of grammars and their parsers. Some of them are discussed in Transformational grammar. A few general remarks are given below.

TG makes central use of a base phrase-structure-grammar component (usually a CFG) to specify a set of base trees (deep structures) and a transformational component to map those trees into a set of surface-structure trees. A simple operation (usually just extracting the terminals) on a surface-structure tree yields one of the sentences in the language thus specified. The meaning of a sentence is encoded in either the base structures, the surface structures, or some combination of both, depending on the type of TG in question.

The transformational component consists of a set of one or more transformations, usually ordered in their application in a rather complex way. A transformation maps each of a class of trees that satisfies certain conditions into a corresponding tree.

Note that the normal direction in which transformations are formulated to operate is from deep structure to surface structure. In transformational parsing, however, one is required to find the corresponding deep and surface structures in a sentence. For some applications and some variants of TG it is sufficient to find only the surface structure or only the deep structure, but it is in general only possible to know one of them has been correctly determined if the other has also been determined and the transformational mapping between them has been verified.

One way to determine the deep and surface structures assigned by some TG to a given sentence is to limit the possible phrase-structure and transformational rules that might be applicable to the derivation of that sentence and then to try all combinations of these rules in the forward direction to see which paths terminate with the given sentence. This is called analysis by synthesis. It was suggested by Matthews (20) but was never implemented. It appears to be prohibitively inefficient.

Another way to determine deep and surface structure is to reverse the forward generative procedure, going from a sentence to its surface structures. Unfortunately, the machinery for forward generation is not directly convertible to a corresponding system for going in the other direction. The first step, determination of the surface structure corresponding to a given sentence, is complicated by the fact that it includes structure reflecting base phrase-structure productions as well as other structure of transformational origin. Petrick (21) has shown that it is possible to compute from a given TG meeting certain requirements a new CF grammar whose productions are a superset of the TG's base component productions and that generates a set of structural description trees that include all of those surface structures assigned to sentences of length not exceeding  $n$  by the TG. This augmented base-component CF grammar can be used to determine all of the surface structures a TG assigns to any sentence of length  $n$  or less, together with some possible spurious ones (given a particular sentence of some length, the augmented CF grammar valid for sentences up to that length can be found if one has not already been determined).

Petrick also presents several ways of inverting the effect of transformations. True inverse transformations are, in general, not computable, but pseudoinverse transforms can be mechanically computed from a given set of forward transforms, and they can be applied, together with some additional CF parsing, to obtain a set of structures that includes all of the deep structures assigned by the TG. These must be checked to ensure that they contain only base-component phrase structure and to ensure that they may be mapped into the previously determined surface structure using the transformational component of the TG in question.

This parsing algorithm is dependent on certain restrictions being placed on the class of TGs to which it is applied. Without such restrictions, classes of grammars such as those of Chomsky's aspects model have been proved to be equivalent to Turing machines (qv) (22), and hence it is known that no parser valid for the entire class is possible.

A final point to note with respect to transformational parsing is that most parsers labeled transformational are not constructed from a given TG in such a way as to guarantee their correctness. Neither are they usually constructed by hand and then proved to be valid parsers of normally formulated TGs.



Rather, they are usually mechanisms of some kind that directly produce structures of the type thought to be correct at some time by the advocates of some variant of TG. Examples of this type that make use of inverse transformations include parsers developed at MITRE (23) and IBM (24,25). Examples that do not make any use of transformational but do attempt to produce "transformational" structure include certain ATN parsers and Marcus parsers (26).

### Augmented-Transition-Network Parsing

Augmented transition networks (ATNs) have played a major role in computational linguistics (qv), since the early 1970s, providing syntactic analysis for many systems with natural-language-understanding capabilities (see Grammar, augmented-transition-network). They are a natural extension of finite-state automata, the automata equivalent of finite-state grammars.

A finite-state automaton (FSA) is a finite set of states connected by directed arcs, each labeled with a symbol from a terminal vocabulary  $V_T$ . One state is designated as the initial state, and some subset of the states are designated as final states. In following a path from the initial state to any of the final states, a sequence of arcs is traversed, and the corresponding string of labels from  $V_T$  is said to constitute a sentence specified by the FSA. The set of sentences so specified is the language generated by the FSA. The automaton is said to be a deterministic finite-state automaton (DFA) if no node has two or more outgoing arcs bearing the same label. Otherwise, the automaton is said to be a nondeterministic finite-state automaton (NFA). There is an effective procedure for constructing from an NFA an equivalent minimal state DFA, i.e., a DFA that generates the same language and does so with the use of the smallest possible number of states.

One improvement of this model of language is the replacement of specific natural-language words as members of  $V_T$  by categories and use of a conventional lexicon to assign specific words to these categories. A further improvement is a generalization of the NFA, namely the basic transition network (BTN), which is weakly equivalent to CF grammars. BTN's contain several different kinds of arcs, most of which are not of major concern. Two of note, however, are the CAT (category) arc and the PUSH arc. CAT arcs are merely ones with labels denoting word categories—parts of speech to be augmented by a lexicon as already discussed above. PUSH arcs are the basic recursion mechanism that extends finite-state grammar equivalency to context-free grammar equivalency. PUSH arcs contain the name of another basic transition network. To traverse a PUSH arc labeled A, control is transferred to the transition network named A (the convention generally used has a transition network named by its initial state), and when it reaches one of its final states, control is transferred back to the state pointed to by the push arc labeled A. Corresponding to the previously given grammar G1 is the BTN shown in Figure 2.

To illustrate the manner in which BTN parsing proceeds, consider the parsing of the sentence "I saw a man in the park" with respect to this BTN: Beginning in state S/ one must traverse the arc labeled PUSH NP/. Beginning at state NP/ of the subnetwork with initial state NP/, one finds three choices. In this exposition one set of correct choices are made, but exhaustive nondeterministic following of all paths is required to

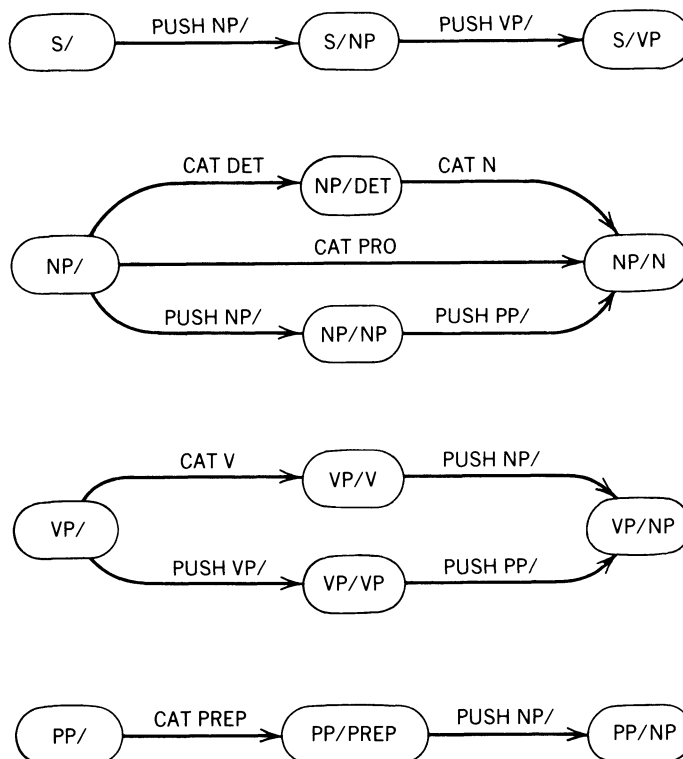


Figure 2.

guarantee finding all structural descriptions. One can take the arc labeled CAT PRO because the next input-string word "I" is of category PRO. This leads to the final state NP/N, completing the traversal of this network and returning control back to state S/NP, the state pointed to by the arc labeled PUSH NP/. Now the arc labeled PUSH VP/ is traversed. Thus, control is transferred to the subnetwork with initial state VP/. Both arcs out of this state lead to acceptance, but only the PUSH VP/ arc will be followed. If once again in state VP/ the CAT V arc is followed, the next input symbol "saw" is consumed, and one is taken to state VP/V. The arc out of this state is labeled PUSH NP/, and this leads to state NP/. From there one can traverse through state NP/DET to the final state NP/N, consuming in the process the next two input symbols, "the" and "man." From this final state NP/N, control is passed back to state VP/NP, which is also a final state. This, in turn, transfers control back to state VP/VP because the PUSH VP/ condition has been satisfied. One next encounters the arc PUSH PP/ and pushes to the subnetwork with initial state PP/. The arc labeled CAT PREP is traversed, consuming the next input string symbol "in," placing one in state PP/PREP. The only arc out of this state is labeled PUSH NP/, and one is able to satisfy this arc by successfully traversing the subnetwork named NP/, consuming in the process the last two input string symbols "the" and "park." This puts one in the final state PP/NP, which satisfies the PUSH PP/ arc and takes one to state VP/NP. It is also a final state, so control is transferred back to state S/VP because the arc labeled PUSH VP/ has been satisfied. The sentence is accepted because one ended in a final state of the top-level subnetwork and consumed in the process the entire input string of words.

It is seen that the BTN formalism is one means of specifying a context-free parser. By adding the use of a set of registers

and extending the types of arcs to include register testing and setting, by allowing other types of conditions and actions to be associated with arc traversal, and by providing for sentence structure to be built up and stored in registers, the popular ATN formalism, due to Woods (27), is defined.

ATNs are equivalent in their generative power to unrestricted rewriting systems or Turing machines, a circumstance that indicates that further restrictions are needed to obtain a model appropriate for the specification of natural languages. Without such restrictions, ATNs are not even guaranteed to terminate.

Note in the illustration using the BTN parser for the language of grammar G1 that parsing was performed in a top-down, left-to-right fashion. This is the natural way to parse using an ATN, but with its Turing machine power, it is not the only way. Note also that, as formulated here, recursive left branching would lead to a nonterminating sequence of actions being taken. To rectify this, one must either disallow networks that reflect recursive left branching or restrict the PUSH and POP mechanism to limit the depth of PUSHing.

The popularity of ATNs in computational linguistic circles has already been noted (e.g., see Ref. 28), and it is reasonable to ask why this is so. The choice does not appear to be motivated from linguistic considerations. See Ref. 29 for a discussion of this. ATN parsers are, however, easier to implement than transformational parsers. The often made, but unsupported, claims of greater speed, which are discussed in the next section, are also probably responsible for the choice of an ATN in many applications. One advantage that TG might have been expected to have over ATNs was the existence of many grammars or grammar fragments, especially in the early days of ATN development. This advantage was not important, however, because very few transformational grammars of any size have been produced. Linguists have been more concerned with refining the type of transformations and the conditions on their usage allowed than with writing and testing large, coherent grammars. Similarly, one could ask why the ATN formalism is being replaced in large measure by the phrase-structure grammar-based models of language and parsing cited above. Linguistic arguments for some of these models have been made, earlier criticism of them has been refuted, and some success in making practical application of them has been achieved. Also, progress in semantics has made the semantic interpretation of the structural descriptions they assign more tenable. Most important of all, at least some of them are basically simpler models and, as such, will remain of interest until they can be shown to have inherent defects or until another unquestionably superior model emerges.

### Empirical Comparisons of Parsing Speed

A few results relating to known mathematical bounds on some of the parsing methods described have been cited above. Results relative to the worst-case performance of the parser for a class of grammars, however, are of less importance than the performance of a particular parsing algorithm on specific grammars of actual interest for some application. Hence, empirical results of parsing efficiency are, for some purposes, more important than theoretical results.

There are, however, difficulties in making empirical comparisons of parsing speed. Allowance must be made for differences in computer speed, for differences in efficiency of imple-

mentation (programming), for differences in direct parsing time vs. indirect time requirements (paging, I/O, garbage collection, etc.), and for differences in coverage of a natural language provided by the grammars with respect to which parsing is performed. Nevertheless, a few results are cited to illustrate the disparity often encountered between theoretical and empirical comparisons of efficiency.

One study of this type is due to Slocum (30). He gives experimental results comparing the efficiency of a Cocke, Kasami, and Younger (CKY) CF grammar parsing-algorithm implementation with that of a left-corner parsing-algorithm implementation. These results were obtained using a large CF grammar for a subset of German. He considered a number of factors, some of which relate to the use of complex feature restrictions, but his basic conclusion was that the left-corner parser was 7% faster than the CKY parser for the sentences considered, even though the former had a worst-case exponential bound and the latter only an  $n$ -cubed bound. Slocum cites another study in which a recursive left-descent parser was compared to two different versions of CKY parsers, two different database-query grammars being used in this study. Both of the CKY parsers proved faster than the top-down parser, by factors of roughly 2–5, depending on the grammar and parser.

A similar study was made by this author in comparing the left-corner parser used in the TQA system with the Ruzzo chart parser described above. The two LISP implementations in question were comparable with respect to their care in choosing and manipulating appropriate data structures, and the same computer and LISP system were used for both. In this study 45 sentences from a database-retrieval application were chosen for comparison. These were sentences that had been submitted to the TQA system by city employees while the system had been installed at the city hall of a local municipality. The grammar in question was a large CF grammar used to assign surface structure to sentences as one step in transformational parsing. For 78% of these sentences the left-corner algorithm was faster by an average factor of more than 2, and for the remaining 22% the Ruzzo parser was faster. The latter sentences, however, were the ones with significantly greater parsing times, resulting in overall lower average parsing time for the Ruzzo parser (0.78 s/sentence) than the left-corner parser (1.15 s/sentence).

Finally, another empirical study of parsing efficiency is a comparison of parsing efficiency for an ATN parser and a TG parser. Petrick (31) compared the ATN LUNAR system parser to that of the TQA system and concluded that although the former's syntactic component was faster, it achieved this by assigning structural descriptions that were less satisfactory for subsequent semantic processing (translation to a formal query language). Prepositional phrases, for example, were simply attached to the nearest noun phrase rather than to the noun phrase they actually modified. TQA system structural descriptions, on the other hand, contain no prepositional phrases. They are reduced to noun-phrase arguments of abstract verbs. For this reason, the LUNAR system required more extensive and time-consuming semantic processing to translate structural descriptions to machine-interpretable form, and the resulting overall syntactic plus semantic processing times of the two systems were found to be comparable.

This study illustrates some of the difficulties that must be faced in making empirical comparisons. Different grammars were involved, and the differences in coverage of English they offered had to be estimated. The quality of the structural de-

scriptions they assigned also had to be taken into consideration because simplifying the work of the syntactic component as found to complicate that of the semantic component. And finally, different computers and programs were used. Unfortunately, such disparities are all too often encountered in making empirical comparisons of diverse methods of parsing.

## Discussion

As has been indicated, there are a host of parsers based on almost as many different linguistic theories and their associated grammars. The requirement of an efficient parser is just one of the demands placed on a class of grammars, so the popularity of particular parsers at different times has to some extent reflected the favored theories of language at those points but has often deviated. In recent years computational linguistics (qv) in general and parsing in particular have exerted more influence on the development of theoretical linguistic models than was previously the case.

The sequence of events in the development of parsing has been something like the following: Early efforts in machine translation were based on no syntactic theory beyond part of speech classification and the identification of a few strings of words and word categories. In the sixties efforts were made to develop CF natural-language grammars and their associated parsers. This gradually fell into disfavor because the CF model was generally discredited by linguists, the semantic interpretation of structural descriptions assigned by CF grammars presented problems that seemed to limit the value of the CF model, and empirical results involving the parsing of sentences with respect to large CF grammars indicated it was not possible to achieve wide coverage of a natural language without assigning a very large (more than 100 in many cases) number of structural descriptions, whose indications of sentence meanings were, at best, tenuous (31).

For these reasons, efforts were made beginning in the mid-sixties to develop transformational grammar parsers to complement the dominant linguistic theory, TG. ATNs, beginning in the early seventies, were motivated in large part by the desire for an efficient natural-language parser and subsequently have been espoused and modified by some theoretical linguists. Both ATN and TG parsers continue in usage to date.

In the eighties there is a resurgence of interest in phrase-structure grammars, usually augmented with complex features and other extensions. It is natural to ask why this has happened. Returning to the reasons cited above for the earlier demise of the CF model, many of the previously given theoretical arguments against CF grammars were successfully refuted (particularly those involving subsets of natural languages that are not CF), and much progress was made on the semantic interpretation of CF grammar-assigned structural descriptions. The remaining objection, wide coverage without wild ambiguity of structural description assignment, remains to be satisfied. No comprehensive grammars of large natural-language subsets have yet been produced and subjected to parsing experiments designed to determine whether this objection has been overcome. Clearly, it is the hope of the developers of phrase-structure-based systems that their augmentations of CF theory and their newly developed innovations in writing grammars for those systems will prove adequate in providing just those structural descriptions that reflect all required meanings and no others.

## BIBLIOGRAPHY

1. A. V. Aho and J. D. Ullman, *Theory of Parsing, Translation and Compiling*, Vol. 1, *Parsing*, Prentice-Hall, Englewood Cliffs, NJ, 1972.
2. J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.
3. A. V. Aho and J. D. Ullman, *Principles of Compiler Design*, Addison-Wesley, Reading, MA, 1977.
4. T. V. Griffiths and S. R. Petrick, "On the relative efficiencies of context-free grammar recognizers," *CACM* 8(5), 289-300 (May 1965).
5. D. J. Rosenkrantz, "Matrix equations and normal forms for context-free grammars," *JACM* 14(3), 501-507 (1967).
6. S. Warshall, "A theorem on Boolean matrices," *JACM* 9, 11 (January 1962).
7. T. V. Griffiths and S. R. Petrick, Top-down versus Bottom-up Analysis, *Proceedings of the IFIP Congress 68*, Edinburgh, UK, *Information Processing 68*, pp. 437-443, 1969, North-Holland, Amsterdam.
8. T. V. Griffiths and S. R. Petrick, Relative Bounds for Two Methods of Context-Free Grammar Parsing, IBM RC 2699, November 1969.
9. D. H. Younger, "Recognition and parsing of context-free languages in time  $n^3$ ," *Inf. Ctrl.* 10, 189-208 (1967).
10. M. Kay, Experiments with a Powerful Parser, *Proceedings of the Second International Conference on Computational Linguistics*, Grenoble, August 1967.
11. J. Earley, "An efficient context-free parsing algorithm," *CACM* 13(2), 94-102 (Feb. 1970).
12. R. M. Kaplan, *A General Syntactic Processor*, Algorithmics, New York, 1973.
13. W. L. Ruzzo, S. L. Graham, and M. A. Harrison, "An improved context-free recognizer," *ACM Trans. Progr. Lang. Sys.* 3, 415-462 (July 1980).
14. G. Gazdar, E. Klein, G. K. Pullum, and I. A. Sag, *Generalized Phrase Structure Grammar*, Blackwell, Oxford, UK, and Harvard University Press, Cambridge, MA, 1985.
15. F. C. N. Pereira and D. H. D. Warren, "Definite clause grammars for language analysis: A survey of the formalism and a comparison with augmented transition networks," *Artif. Intell.* 13, 231-278 (1980).
16. M. Kay, Parsing in Functional Unification Grammar, in *Studies in Natural Language Processing*, Cambridge University Press, Cambridge, UK, pp. 251-278, 1985.
17. E. Proudan and C. Pollard, Parsing Head-Driven Phrase Structure Grammar, in *Proceedings of the Twenty-Third Annual Meeting of the Association for Computational Linguistics*, University of Chicago, Chicago, IL, pp. 8-12, July 1985.
18. J. Bresnan (ed.), *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, MA, 1982.
19. S. M. Shieber, H. Uszkoreit, F. C. N. Pereira, J. J. Robinson, and M. Tyson, *The Formalism and Implementation of PATR-II, Research on Interactive Acquisition and Use of Knowledge*, Artificial Intelligence Center, SRI International, Menlo Park, CA, 1983.
20. G. H. Matthews, Analysis by Synthesis of Sentences of Natural Languages, *Proceedings of the 1961 International Congress on Machine Translation of Languages and Applied Language Analysis*, Teddington, U.K., National Physical Laboratory, 1961.
21. S. R. Petrick, *A Recognition Procedure for Transformational Grammars*, Ph.D. Dissertation, MIT, Cambridge, MA, 1965.
22. P. S. Peters and R. W. Ritchie, "On the generative power of transformational grammars," *Inf. Sci.* 6, 49-83 (1973).
23. A. Zwicky, J. Friedman, B. Hall, and D. Walker, The MITRE

- Syntactic Analysis Procedure for Transformational Grammars, *Proceedings of the 1965 Fall Joint Computer Conference*, Thompson, Washington, DC, 1965.
24. S. R. Petrick, Transformational Analysis, in R. Rustin (ed.), *Natural Language Processing*, Algorithmics, New York, pp. 27–41, 1973.
  25. D. G. Loveman, J. A. Moyne, and R. G. Tobey, CUE: A Preprocessor System for Restricted Natural English, in J. Minker and S. Rosenfeld (eds.), *Proceedings of the Symposium on Information Science and Retrieval*, College Park, MD, University of Maryland, April 1971.
  26. M. Marcus, *A Theory of Syntactic Recognition for Natural Language*, MIT Press, Cambridge, MA, 1980.
  27. W. A. Woods, "Transition network grammars for natural language analysis," *CACM* 13(10), 591–606 (October 1970).
  28. D. L. Waltz (ed.), "Natural language interfaces," *ACM SIGART Newslett.* (61), 16–64 (February 1977).
  29. B. E. Dresher and N. Hornstein, "On some supposed contributions of artificial intelligence to the scientific study of language," *Cognition* 4, 321–398 (1976).
  30. J. Slocum, A Practical Comparison of Parsing Strategies, *Proceedings of the Nineteenth Annual Meeting of the ACL*, Stanford University, Stanford, CA, June 29–July 1, 1981, pp. 1–6.
  31. S. Kuno, The Current Grammar for the Multiple Path English Analyzer, *Mathematical Linguistics and Automatic Translation*, Report No. NSF 8, Computation Laboratory, Harvard University, Cambridge, MA, 1963.

S. PETRICK  
IBM

PARSING, CHART. See Parsing.

## PARSING, EXPECTATION-DRIVEN

### Basic Characteristics

The term "expectation-driven parsing" refers to a family of natural-language understanding (qv) systems that share the following features:

"parsing" is defined to mean the production of conceptual meaning structures, not syntactic parse trees (See Parsing); parsing is done left to right, and what is parsed first leads to expectations about what will be parsed next; parsing information is attached to words and concepts rather than to patterns of syntactic categories; parsing information is represented with if-then rules, called requests, which are used in a forward-chaining manner to "expect" or "predict" what concepts will appear in the text, and requests may look for such syntactic cues as word order and phrase and clausal boundaries, but normally this does not lead to the construction and later use of syntactic structures larger than noun phrases.

An expectation-driven parser, in other words, is a top-down parser that looks for concepts rather than grammatical elements. So, for example, where a syntactic parser might look for a verb phrase followed by a noun phrase, an expectation-driven parser would look for an action followed by an object

that the action could apply to. The availability of conceptual structures during the parsing process is the most distinctive feature of expectation-driven parsers. Other parsing systems, such as augmented-transition-network parsers (see Grammar, augmented-transition-network), use similar if-then control rules, but they use them to build syntactic structures, and semantics is present, if at all, primarily to filter out unlikely syntactic combinations. With expectation-driven parsers, it is just the other way around; syntactic cues help to select reasonable conceptual combinations.

A request has at least two parts: a test and an action. The basic expectation-driven parsing algorithm, which each theory extended in some direction, uses a list of active requests, initially empty, and a set of global variables, where the actions of the requests can put conceptual structures (see Conceptual dependency). One variable, call it CONCEPT, is used to hold the conceptual structure for the whole sentence. The basic algorithm is given below.

1. Read a word (going from left to right). If there are no more words, stop and return whatever is in CONCEPT as the parse of the sentence.
2. Otherwise, add the requests attached to that word to a list of active requests.
3. Perform the actions specified by every request in the active list that has a test that evaluates to true. This is called triggering and executing requests.
4. Go back to step 1.

The development of expectation-driven parsers were directed by the needs of larger projects that required conceptual interpretations of particular input texts. Each new project set new challenges, requiring more complex input texts and richer output representations. A proper description of any particular expectation-driven parser needs to include the kinds of texts it was intended to handle and the kind of system it was intended to talk to.

### Beginning

The notion of expectation as a key element in conceptual analysis is presented in Ref. 1 and expanded in Ref. 2. Expectations could occur at six different levels, ranging from the syntactic; through the conceptual, and up to the life-death continuum of the memory model.

The first request-based expectation-driven parser was developed for the MARGIE inference system (see Inference) (3,4). At that time, expectation and its usefulness to parsing were defined thus:

*An expectation is a description of a situation that is likely to become true in the near future. Further, associated with any expectation is a set of actions to perform, actions that are appropriate in the expected situation. . . . The importance of an expectation is not just that it prepares a set of actions for use if needed, but also that it narrows how future situations are perceived. An expectation looks only for certain features and ignores any other features of the real situation that may be present (5).*

The MARGIE system (6) modeled reasoning chains about everyday human actions, such as:

being hit hurts;  
doing an action implies you wanted the results; and  
ergo, if John hits Mary, he wanted Mary to be hurt.

MARGIE used the conceptual-dependency (CD) knowledge-representation scheme, which has about a dozen canonical primitive acts that represent events and causal sequences, including motion (PTRANS), transfer of possession (ATRANS), communication (MTRANS), and application of force (PROPEL). Each CD action is a frame with slots for actor, object, etc.

The goal of the MARGIE parser (TMP) was to convert English sentences, such as "John gave Mary a kiss" or "John prevented Bill from taking the bike by giving the bike to Mary," into CD forms that the MARGIE reasoner could use.

Following the parsing algorithm given above, the MARGIE parser read a sentence from left to right, word by word, putting the requests attached to each word in a list, and executing the actions of any requests whose tests were true. Requests that were executed were removed from the list. Requests that were not executed were kept in the list and checked again each time a new word was read. Usually, the basic conceptual framework for the sentence was built as soon as the main verb's requests were loaded, and the remainder of the sentence processing involved filling in the slots.

Figure 1 shows a slightly simplified definition of "give" (7). The English equivalent of these requests is given below.

Always make TO1 (which looks for "to someone") the preferred sense of "to."

If you see a human, save it in the RECIP register.

If you see a physical object, save it in the OBJ register.

If you see a physical object, set the meaning of the sentence to a "transfer" action, where the SUBJECT is giving the OBJ to the RECIP.

If you see a description of an action that takes an object, set the meaning of the sentence to that action, where the SUBJ is doing the action to the RECIP.

The last request was for sentences such as "John gave Mary a kiss," producing exactly the same result as "John kissed Mary." Most parsers would produce, at best, something equivalent to "GIVE-ACTION(JOHN, MARY, KISS)" and leave it to the inference machinery to figure out that this meant the same thing as "KISS(JOHN, MARY)." The MARGIE parser

always produced the correct final CD rather than an intermediate form.

One important class of requests in TMP modified the list of active requests. In particular, the requests attached to the word "a" hid the current set of requests and added a new request whose test looked for the end of the noun phrase and whose action built a CD form representing the meaning of the noun phrase and restored the hidden set of requests.

By hiding the current set of requests, the pieces of a noun phrase could be put together without accidentally triggering requests set up by other words in the sentence.

## ELI

The successor to the MARGIE parser was ELI (qv) (English Language Interpreter) (8), part of the SAM (qv) (Script Applying Mechanism) (9) system. The goal of the SAM system is to

read short texts (half a dozen lines or so) about events occurring in a stereotypical situation, e.g., a story about going to a restaurant or a newspaper report on a car accident; fill in those events that most people would assume must have happened, even though they were not mentioned; and answer questions about what did and did not occur in the story.

The goal of ELI is to take these input texts and assign CD interpretations to them that SAM could use.

ELI made more explicit the functional structure of requests. A new field was added to each request, called SUGGESTIONS, that contained requests to activate if the first request was executed. Requests adding requests was much more common in ELI than in the MARGIE parser. Nesting requests in this way reduced the number of requests active at any one time and increased the "expectational" nature of ELI.

ELI also made explicit in each request what variables its test depended on and what variables its action affected. By doing this, it was possible to dynamically chain requests together, resulting in several improvements in request management:

when the value of a global variable changed, ELI could immediately tell which tests of which requests might be affected;

a request whose action would set a variable to a value that no test wanted could be rejected immediately;

default constraints placed on variables could be propagated automatically to those requests whose actions were attempting to fill those variables; and

once a variable was filled, other requests attempting to fill that same variable could be removed from the list of active requests as no longer needed.

Figure 2 shows one of the requests for "give" (10). In English, this says:

always return an ATRANS structure, where the ACTOR is the source (FROM) of the transfer, the ACTOR and recipient (TO) are human, and the OBJECT is a physical object, and

add two requests. The first one tests the subject concept variable (CONTOPIC) and returns it. The second tests the input concept variable (CONINFO) and returns it.

TEST	ACTION
Always true	Set lexicon entry for 'TO' = 'TO1'
SENSE is human?	Set RECIP = SENSE
SENSE is physical object?	Set OBJ = SENSE
SENSE is physical object?	Set CONCEPT = 'ATRANS <u>frame</u> '
	Set ACTOR = SUBJ
	Set OBJECT = OBJ
	Set TO = RECIP
	Set FROM = SUBJECT
SENSE is action?	Set CONCEPT = SENSE
	Set ACTOR = SUBJ
	Set OBJECT = RECIP

NOTE: SENSE is a variable set to a conceptual meaning of the current input.

Figure 1. Definition of "give" in MARGIE parser.

```

TEST FOCUS: None
TEST PREDICATE: Always true
ACTION: 'ATRANS frame'
MODIFICATIONS: ACTOR = FROM
                  ACTOR is human
                  OBJECT is physical object
                  TO is human
SUGGESTIONS:
1) SUGGESTION FOCUS: ACTOR
   TEST FOCUS: CONTOPIC
   TEST PREDICATE: Always true
   ACTION: CONTOPIC
2) SUGGESTION FOCUS: TO
   TEST FOCUS: CONINFO
   TEST PREDICATE: Always true
   ACTION: CONINFO

```

Figure 2. ELI request for "give."

As in the MARGIE parser, there is another request (not shown) to handle sentences with direct objects describing actions, as in "John gave Mary a kiss."

The requests for ACTOR and TO did not have to say explicitly what kind of concept they were looking for because the constraints on the ACTOR and TO slots were propagated automatically. Thus, the first added request is effectively testing for something in CONTOPIC that is human because that is the constraint propagated from the ACTOR slot of the ATRANS.

This automatic propagation of constraints (see Constraint satisfaction) made it possible for ELI to convert general predictions from SAM to specific parsing predictions. For example, after "The waiter brought Mary a menu," SAM predicted that Mary would MTRANS what she wanted to eat to the waiter. If ELI then read "Mary gave . . .," it would reject the normal ATRANS request for "give" and predict that the direct object of "gave" should be a noun phrase meaning MTRANS, as in "Mary gave her order." Unfortunately, this intriguing aspect of ELI was never developed past the experimental stage.

## ELI-2

The successor to ELI was ELI-2 (11), which was used with SAM when that system was adapted to read real newspaper texts and headlines. Instead of sentences like "When the check came, John paid and left a tip," where clausal relationships are most important, newspapers have sentences like "John Doe, 27, of 905 Foxon Highway, the driver of the vehicle, was killed when his car swerved off a road and struck a tree, police reported." Multiple appositive noun phrases, prepositional phrases, and special forms such as "police reported" did not fit the purely expectational ELI model.

Several changes were made to repair these deficiencies in ELI. First, ELI-2 had three different modes of operation, depending on the local linguistic situation:

- expectation-driven, as in ELI, where each new concept from the text was immediately passed to the list of active requests;
- position-driven, inside noun groups, where concept merging was held until the end of noun phrases; and
- situation-driven, for "police reported," prepositional phrases, etc., where the new concept "inserted itself" into the current main conceptual form rather than being picked up by some existing request.

Furthermore, ELI's simple set of active requests was an ordered list of sublists in ELI-2:

- the WORD list held requests for specific words or word features;
- the ACTIVE list held requests for specific concepts;
- the OPTIONAL list held requests for concepts that could, but did not have to, appear, e.g., "John went to Chicago from Boston";
- the EXPERT list held general grammatical knowledge, e.g., about relative clauses and prepositional phrases;
- the END-OF-PHRASE list held requests triggered by completion of phrases; and
- the CONTINGENCY list held requests used to recover from uncertain choices about ambiguous words.

Finally, ELI-2 generalized the notion of the current concept variable (SENSE in the MARGIE parser, CONINFO in ELI) to a list of recently built concepts. In this way several conceptual structures could be built and held before being joined into one main structure.

## CA

ELI-2 was succeeded by the Conceptual Analyzer (CA) (12). CA's control structure was simpler than ELI-2's, so that people developing particular reasoning modules could easily extend CA to parse the texts appropriate for their domains.

Like ELI, CA hid requests when parsing noun phrases, and like ELI-2, CA kept a list of recently built concepts, called the C-LIST, from which larger structures were built. Requests could not only test for a particular concept in the C-LIST but could also test the order in which concepts appeared in the C-LIST. This merged ELI-2's situation-driven and position-driven modes into one framework.

CA's active request list was subdivided into *pools*, where each pool contained those requests that had been added at the same time. CA always used the most recent requests first, so that a loose stack discipline was maintained for reducing the number of requests considered.

An example sentence that CA dealt with was "A small plane stuffed with 1500 pounds of marijuana crashed 10 miles south of here." Figure 3 shows the CA definition, in pseudo-code form, for the word "stuffed" (13). The local variable RESULT is used by the second subrequest to make a CD with a self-reference, representing "A small plane, where someone

```

TEST: Always true
ACTIONS:
  Set RESULT = 'PTRANS TO INSIDE frame'
  Put RESULT on C-LIST
  Activate:
    TEST: Next word = 'WITH'?
    ACTIONS: Activate, in the next pool, high priority:
      1) TEST: There's an object X on C-LIST,
         and TO-INSIDE of RESULT is filled
         ACTIONS: Set OBJECT of RESULT = X
      2) TEST: There's an object X on C-LIST
         preceding RESULT
         ACTIONS: Set REL of X = RESULT
                  Set TO-INSIDE of RESULT = X

```

Figure 3. CA request for "stuffed."



stuffed *that plane* with. . . ." The two subrequests would be put into the same pool and added to the front of the request list.

## AD-HAC

Two major problems with the MARGIE parser's control structure were that old requests piled up because they could only be removed by successfully firing, and requests that did very different kinds of actions were lumped together indiscriminately. ELI attacked the first problem by attaching requests to the variables they affected, and ELI-2 attacked the second by organizing requests into functionally distinct sublists.

The independently developed AD-HAC analyzer (14) proposed some alternative solutions. "Keep predicates" were attached to requests to control when unused requests should be removed, and requests were organized into request classes. These classes were used to control the order of the application of requests in the formation of theories, where a theory consisted of a partially built conceptual parse, a numeric measure of goodness for this structure, a set of still unused constituents, and a set of requests looking for particular constituents to complete the partial conceptual parse. Different theories were constructed as requests made different choices about which constituent should go where in the parse, and the goodness rating was used to pick the best theory.

Like the Word-Expert Parser described below, AD-HAC was developed to deal with very ambiguous sentences using a preferential approach (15). AD-HAC's theories were a way of exploring possible conceptual parses in a best-first manner. One very novel feature of AD-HAC was that it used an ATN syntactic parser to preanalyze the input texts and label verb groups, noun groups, prepositional phrases, and conjunctions, thereby removing the need for requests attached to articles and other function words. The output of this preanalysis bracketing preserved most of the interesting ambiguities for conceptual analysis. For example, "her money" was left open as potentially one noun phrase or two.

## Micro ELI

Micro ELI (16) was developed to teach the basic principles of expectation-driven parsing. Micro ELI's requests had three fields: TEST, ASSIGN, and NEXT-PACKET. The ASSIGN field reduced all actions to variable assignment. \*CD-FORM\* contained the current concept, \*CONCEPT\* contained the CD for the whole sentence, etc. The NEXT-PACKET field of a request contained requests to be added if the request was executed. As in CA, requests were added in separate pools rather than merged into one list. Finally, Micro ELI used a very simple, strict stack discipline: the most recently added packet of requests had to be used or removed before the next most recent packet could be accessed.

Figure 4 shows the definition of the word "got" (17).

## Problems and Solutions

The MARGIE parser, ELI, ELI-2, and CA all used lexically indexed requests to construct conceptual forms with only simple syntactic cues. Several other systems, however, although preserving the primary goal of producing conceptual representations, proposed significant alternatives in control structure.

```

ASSIGN: Set *PART-OF-SPEECH* = 'VERB
        Set *CD-FORM* = 'ATRANS frame'
        Set ACTOR = *SUBJECT*
NEXT-PACKET:
        TEST: *PART-OF-SPEECH* = 'NOUN-PHRASE?'
        ASSIGN: Set OBJECT = *CD-FORM*

```

Figure 4. Definition of "get" in Micro ELI.

**Word-Expert Parsing.** One of the first alternatives developed was Small and Rieger's Word-Expert Parser (WEP) (see Parsing, word-expert) (18). The key problem WEP was designed to solve was lexical ambiguity, especially the interaction of several ambiguous words, as in "The philosopher threw the peach pit into the deep pit."

WEP's requests were much larger and more complex than ELI's or CA's. In ELI, for example, the definition for a very ambiguous word, such as "take" or "be", might be a page or so in length, involving a dozen requests, nested two or three levels deep. In WEP the definition of "throw," incomplete though it was, was six pages of code. Much of the complexity arose from the complicated communication protocols needed to allow different words to interact with each other and reach an agreement on the best overall meaning for the sentence. WEP probably pushed the use of requests for doing parse-time inferencing as far as it could go, if not farther.

**IPP (Integrated Partial Parser).** Contrary to WEP, IPP (19) simplified word definitions to the bare conceptual minimum. The most ambiguous words, such as "be" and "take," ended up with no definition at all. Parsing was driven by requests attached to concepts pointed to by more contentful words, such as "hijack" and "shooting." IPP was designed to parse hundreds of newspaper articles about terrorism and store them in a hierarchical long-term database using a memory structure called the memory-organization packet (MOP) (20). With such a large set of inputs, IPP needed both a large basic vocabulary and a robustness in the face of unknown words and unexpected grammatical constructions.

IPP parsing algorithm divided words into the five classes described below.

**Event builders**, e.g., "shot," "hijacker," "killing," which pointed to concepts, called action units (AUs), stored in a large hierarchical conceptual memory. When read, the associated AU was copied (instantiated) from memory and filled out, using the tokens and event refiners found nearby in the text. An AU could also set up requests for certain function words.

**Event refiners**, e.g., "attempted," "fatally," "failed," which were simply saved for processing by an AU.

**Token makers**, e.g., "embassy," "ambassador," "official," which were simply saved until a head noun is seen, at which point a concept token was built and saved for collection by an AU.

**Token refiners**, e.g., "arabic," "left-wing," "twenty-two," which were saved until collected by a token maker.

**Function words**, e.g., "by," "an," "into," which were processed by AU-generated requests.

Most of IPP's parsing involved filling out the action units. For example, "shot" pointed to the AU SHOOT, shown in Figure 5 (21), which was a particular form of the MOP (see Mem-

```

Action unit
TYPE: Script
TEMPLATE: 'SHOOT frame'
ASSOCIATED MOP: 'ATTACK-PERSON'
PASSIVE-USAGE:
1) TEST: SUBJECT is human?
   ACTION: Set VICTIM of SHOOT = SUBJECT
2) TEST: SUBJECT is weapon?
   ACTION: Set WEAPON of SHOOT = SUBJECT

```

Figure 5. Action unit for "shoot" in IPP.

ory-organization packets) personal attack. Note that knowledge about passives, e.g., "The terrorists were shot . . .," was stored with action units rather than individual event-builder words because words pointing to the same AU, such as "shoot" and "fire on," all used the syntactic subject of the passive sentence in the same way, e.g., if the subject was human, the subject was the VICTIM of the shooting.

Because AUs were very often pointed to by nouns, e.g., "hijacking," "explosion," and "kidnapping," IPP was not as verb-centered as most expectation-driven parsers.

**DYPAR/BORIS.** Like ELI and AD-HAC, DYPAR (22) was initially developed to solve the problem of left-over requests. DYPAR's requestlike demons (qv) specified not only when they might be triggered but also when they should be removed. Demons could also activate (spawn) other demons and assign them priorities that determined the order in which they should be tested.

DYPAR changed significantly, however, when it became part of the BORIS (qv) memory-based story-understanding system. DYPAR used BORIS's memory in two ways. First, where ELI and CA would search a parser-maintained concept list, DYPAR's demons searched BORIS's working and long-term episodic memories (qv) (EM and WM). Second, where ELI and CA would build CD structures to pass on, DYPAR instantiated copies of BORIS's memory structures.

Figure 6 shows the BORIS memory structure for "writing a letter" pointed to by one sense of the word "write" (23). Each slot in the memory structure had one or more demons for filling that slot. The one attached to LETTER-READER waited for a human to be added to working memory, perhaps preceded by the word "to." Instantiating the MOP involved searching long-term episodic memory for an event matching the one in working memory. If one could not be found, a new one was created and added to memory.

**MOPTRANS.** With the development of DYPAR and IPP, two things about expectation-driven parsing had become clear. First, the goal of producing conceptual representations during the parse had become one of accessing and instantiating long-term memory structures. Second, every system, from the MARGIE parser up through DYPAR, suffered from a great deal of redundancy in the system. The same requests appeared

```

MOP: WRITE-LETTER
  Fill LETTER-WRITER with preceding human
  Fill LETTER-READER with succeeding human, or
                           human preceded by 'TO'
  Fill LETTER-INFO with ...
  Fill EVENT with 'MTRANS BY LETTER frame'
  Instantiate in long-term memory

```

Figure 6. Memory structure for "write" in DYPAR.

in many different definitions. For example, a large number of verbs had a request that filled the actor of the action with the subject of the sentence.

MOPTRANS (24) was a parser for a machine-translation (qv) system that had to be able to parse several different languages. Both to save space and to keep rules consistent, the basic ELI model was changed in several ways to increase the amount of shared syntactic- and semantic-parsing information, not only between similar words in one language but also between words meaning similar things in the different languages.

First, syntactic knowledge was stored separately from conceptual knowledge, so that one syntactic rule could be shared by many words. It was still true, however, that both kinds of information were applied simultaneously during the parsing process and that syntax was used to help out the basic process of building memory structures.

Second, MOPTRANS was not restricted to lexically indexed rules. It built limited syntactic structures and used them to retrieve generalized syntactic rules, which had the following parts:

- syntactic pattern, which gave a sequence of syntactic objects that must be seen for this rule to be used;
- restrictions, which gave additional tests that have to be true;
- syntactic assignment, which assigned syntactic roles to the syntactic objects;
- semantic action, which built a CD form or filled a slot in an existing form; and
- result, which specified which syntactic objects should remain in working memory and what syntactic class to label them with.

Figure 7 shows two generalized syntactic rules (25).

Finally, conceptual parsing rules, such as how to fill the actor slot of an action, were shared by storing them at the highest level possible in a hierarchical concept memory. Specialization and instantiation rules were used to move from general concepts to more specific ones as the text was read. These specialization and instantiation rules included:

- script-activation rule—if an event in a script has been activated, activate the whole script, e.g., the "capture" event implies the "find" script;
- expected-event specialization rule—if an abstraction of a predicted event is activated, activate the predicted event, e.g., if POLICE-INVESTIGATION is predicted, and "find" is read, POLICE-INVESTIGATION should be activated; and
- slot-filler specialization rule—if a concept's slot is filled with something that normally fills a more specific version of that concept, activate the more specific concept, e.g., "Police are looking for . . ." can be specialized from FIND to the more specific POLICE-INVESTIGATION concept.

## Future

Initially, expectation-driven parsers were developed to construct conceptual meaning structures and pass them on to separate knowledge-based inference processors, such as MARGIE



## SUBJECT rule

PATTERN: NP V  
 RESTRICTIONS: NP not already attached  
 ASSIGNMENT: Set SUBJECT of V = NP  
                   Set MAIN CLAUSE = V  
 SEMANTIC ACTION: Set ACTOR of meaning of V = meaning of NP  
 RESULT: V, relabelled as S

## DATIVE MOVEMENT rule

PATTERN: S NP  
 RESTRICTION: S has no OBJECT, NP free, S allows dative movement  
 ASSIGNMENT: Set INDIRECT OBJECT of S = NP  
 SEMANTIC ACTION: Set RECIPIENT of meaning of S = meaning of NP  
 RESULT: S and NP

Figure 7. Two generalized syntactic rules for MOPTRANS.

and SAM. As theories developed that modeled reasoning as a process of searching and instantiating memory structures (20,26), theories of parsing changed in a similar direction. Increasingly close ties between memory search and parsing can be seen in DYPAR, IPP, and MOPTRANS and are being developed even further (27–29). Expectation-driven parsing did not gradually converge with more standard syntactic approaches to natural-language understanding, as many quite reasonably thought it would. Instead, it merged more and more with the inference processors whose needs it served. Eventually there may be expectation-driven parsers whose requests are general inference rules that produce not conceptual structures but direct pointers to nodes in memory, i.e., there will not be parsers at all, but lexically oriented memory processes.

## BIBLIOGRAPHY

1. R. C. Schank, L. Tesler, and S. Weber, Spinoza II: Conceptual Case-Based Language Analysis, AI Memo 109, Stanford University, Stanford, CA, 1970.
2. R. C. Schank, Intention, Memory, and Computer Understanding, AI Memo 140, Stanford University, 1971.
3. C. K. Riesbeck, Computational Understanding: Analysis of Sentences and Context, Ph.D. Thesis, Stanford University, 1974.
4. C. K. Riesbeck, Conceptual Analysis, in R. C. Schank (ed.), *Conceptual Information Processing*, North Holland/American Elsevier, Amsterdam, pp. 83–156, 1975.
5. Reference 3, p. 20.
6. R. C. Schank, *Conceptual Information Processing*, North Holland/American Elsevier, Amsterdam, 1975.
7. Reference 3, pp. 98–100.
8. C. K. Riesbeck and R. C. Schank, Comprehension by Computer: Expectation-Based Analysis of Sentences in Context, in W. J. M. Levelt and G. B. Flores d'Arcais (eds.), *Studies in the Perception of Language*, Wiley, Chichester, U.K., 1976.
9. R. Schank and R. Abelson, *Scripts, Plans, Goals, and Understanding*, Lawrence Erlbaum, Hillsdale, NJ, 1977.
10. Reference 8, p. 280.
11. A. V. Gershman, Knowledge-Based Parsing, Technical Report 156, Yale University Department of Computer Science, New Haven, CT, 1979.
12. L. Birnbaum and M. Selfridge, Conceptual Analysis of Natural language in R. C. Schank and C. K. Riesbeck (eds.), *Inside Computer Understanding*, Lawrence Erlbaum, Hillsdale, NJ, pp. 318–353, 1981.
13. Reference 12, pp. 345–346.
14. A. Cater, Request-Based Parsing with Low-Level Syntactic Recognition in K. Sparck Jones and Y. Wilks (eds.), *Automatic Natural Language Parsing*, Ellis Horwood, Chichester, pp. 141–147, 1983.
15. Y. Wilks, "A preferential, pattern-matching semantics for natural language understanding," *Artif. Intell.* **6**, 53–74 (1975).
16. R. C. Schank and C. K. Riesbeck, *Inside Computer Understanding*, Lawrence Erlbaum, Hillsdale, NJ, 1981.
17. Reference 16, p. 368.
18. S. Small and C. Rieger, Parsing and Comprehending with Word Experts (A Theory and its Realization), in W. G. Lehnert and M. Ringle (eds.), *Strategies for Natural Language Processing*, Lawrence Erlbaum, Hillsdale, NJ, 1982.
19. M. Lebowitz, "Memory-based parsing," *Artif. Intell.* **21**(4) 363–404 (1983).
20. R. C. Schank, *Dynamic Memory: A Theory of Learning in Computers and People*, Cambridge University Press, New York, 1982.
21. M. Lebowitz, Generalization and Memory in an Integrated Understanding System, Ph.D. Thesis, Yale University, New Haven, CT, October 1980. Also as Research Report 186, p. 266.
22. M. G. Dyer, *In-Depth Understanding*, The MIT Press, Cambridge, MA, 1983.
23. Reference 22, p. 137.
24. S. Lytinen, The Organization of Knowledge in a Multi-Lingual, Integrated Parser, Ph.D. Thesis, Yale University, November 1984. Also as Research Report 340.
25. Reference 24, pp. 129–130.
26. J. A. Feldman and D. Ballard, "Connectionist models and their properties," *Cog. Sci.* **6**(3), 205–254 (1982).
27. S. Small, G. Cottrell, L. Shastri, Toward Connectionist Parsing, *Proceedings of the Second AAAI*, Pittsburgh, PA pp. 247–250, 1982.
28. R. H. Granger, K. P. Eiselt, and J. K. Holbrook, The Parallel Organization of Lexical, Syntactic, and Pragmatic Inference processes, *Proceedings of the First Annual Workshop on Theoretical Issues in Conceptual Information Processing*, Atlanta, GA, pp. 97–106, 1984.
29. C. K. Riesbeck and C. E. Martin, Direct Memory Access Parsing, YALEU/DCS/RR 354, Yale University, New Haven, CT, 1985.

C. K. RIESBECK  
 Yale University

## PARSING, WORD-EXPERT

**Overview: Parsing as a Process.** The theory of word-expert parsing (WEP) treats the individual words of language as the organizational elements of the human knowledge of language interpretation. Each word has a different effect on the understanding process by its range of possible meanings and the discrimination among them and by its interrelationships with other words with which it combines to form larger meaning fragments. Word-expert parsing does not represent a semantic theory (see Semantics) but rather a theory of language analysis (see Natural-language understanding). As such, it describes the representation and use of syntactic, semantic, and real-world knowledge for approximating the cognitive mechanisms of language interpretation.

Word-expert parsing views individual words of language as the fundamental carriers of knowledge about the parsing process. It is crucial to realize the implications of this view to understand the WEP perspective. Individual words to not constitute the input to some mechanism that analyzes sentences

from above according to some general external principles about language. Rather, the words themselves form the mechanism. Each word of language is seen as an active lexical agent called a "word expert," which participates in the overall control of the parsing process by its internal actions and its interactions with other such agents. The meaning of a particular word in some context is dictated by its interrelationships with other words in the text, by the meaning of existing pieces of the text, and by general commonsense knowledge (see Reasoning, commonsense).

The interrelationships among words in a fragment of text ought not to be viewed statically, from some point above where all such associations are perceived as schematic juxtapositions. Words themselves are active agents, idiosyncratically controlling the parsing process. The procedural, distributed way of looking at language leads to the WEP notion of active lexical interaction. From this vantage point, the theory of syntax describes stereotypic patterns of lexical interactions, which, because of the rich semantic particularities of individual words, cannot be used to model comprehension. The process of WEP for a particular fragment of text consists of the active lexical interactions among the individual agents representing the words of that fragment and the interactions between those agents and the other mechanisms of comprehension. The nature and scope of these interactions, and not any particular declarative structure, represent the analysis of the fragment.

The theory of WEP not only provides a different way of looking at language through comprehension but also suggests that such classical problems as word-sense ambiguity and idiom be made the framework of such a theory. Almost every common word of language has many different contextual meanings. Furthermore, the ways in which certain meanings are preferred over others seem to be idiosyncratic to the individual words themselves. The original English dictionary by Samuel Johnson clearly makes (1) this point in describing over a hundred word senses for the word "take" and using each one in a separate example sentence. Certain words can pair up with others in certain ways, with no substitutions possible. Two words meaning similar things can be interchanged in all but a few exceptional cases. Certain sentences can be put in the passive voice and have the same meanings as before, and others cannot. The theory of WEP places the explanatory burden on mechanisms associated with individual words. The exceptional cases are thus promoted to a central place in the theory of comprehension.

The ability for an idiomatic expression to retain its special meaning when manipulated in certain ways has been described (2) as its "degree of tightness." The wide variation in the tightness of particular expressions leads Bolinger to an extreme statement:

*The question arises whether everything we say may be in some degree idiomatic—that is, whether there are affinities among words that continue to reflect the attachments the words had when we learned them, within larger groups. This is not a welcome view to most American linguists.*

The perspective of WEP does not deviate much from the radical postulate suggested here. The individual words of language are perceived as the carriers of parsing knowledge and actively interact with other words, based on idiosyncratic cri-

teria, to create meaningful fragments. Such criteria include notions about particular lexical attachments, general syntax and semantics, and beliefs about the nature of the real world. These criteria are clearly not universal. A single sentence could easily mean two different things to two different people and likewise to the same person at two different times (in the same context). The process of language analysis depends on the knowledge and perspectives of the analyzer.

**Principles.** The control structure of the word-expert parser is intended to suggest some new ideas about the mechanisms of language understanding. Several principles regarding the nature of the understanding process have influenced the organization of the model. Seven of them are described below.

1. Parsing knowledge is organized around individual word experts.
2. Parsing consists of nonuniform word-sense discrimination and active lexical interaction among the distributed word experts.
3. Parsing takes place from left to right, except when directed to do differently by the word experts themselves.
4. Parsing takes place without backtracking, proceeding in a wait-and-see manner (deterministically) toward a meaning interpretation.
5. The reader expectations triggered by particular words in the context of discourse and real-world knowledge comprise an important source of dynamic parsing knowledge.
6. Parsing is principally data-driven, in that these high-level expectations assist, but do not direct, the process.
7. The syntax of language provides structural cues for the interpretation of word meanings and can be viewed as the active lexical interactions among word experts.

These principles combine some new perspectives of the word-based distributed framework with certain perspectives found in previous work, in particular, the works on word-sense selection (3), conceptual analysis and the view of parsing as a memory process (4), and analysis in a data-driven and deterministic manner (5).

**The Word-Expert Parser.** The word-expert parser is a working computer program that analyzes fragments of English text, producing symbolic data structures to represent their unambiguous meaning content in context. The parser successfully analyzes input fragments rich in word-sense ambiguity and idiom and also handles a range of interesting syntactic constructions. The program works by having the individual words of language themselves direct the process. Each word is modeled by a separate computer program (a coroutine) and parsing takes place through the successive execution of the expert programs corresponding to the words of the input.

The effect of these programs is to augment the overall meaning representation and to alter the control flow of the overall process. One problem in reading from left to right is that sometimes the meaning (or role) of a word does not become clear until later in the sentence, after a few more words have been seen. Since each word expert is responsible for fitting its word onto the overall interpretation of the sentence, it sometimes needs to wait a while in just this way, only to continue later. The individual word expert thus affects the high-order

processing of the model by waiting for what it needs and then forcing itself back into the action, obviously disrupting the normal left-to-right flow of things.

The parser was developed in Maryland LISP (qv) on the Univac 1100/42 at the University of Maryland and was converted to run in VLISP on the DEC-10 at IRCAM in Paris and in Franz LISP on the VAX/780 at the University of Rochester. It operates with a small vocabulary of 40 implemented word experts, illustrating an interesting variety of different word-expert functions and interaction requirements. The existing collection of experts is sufficient to analyze sentences containing many different contextual usages of the content words "eat," "deep," "throw," "pit," "case," "by," "in," and "out." The analysis of a sentence containing such a word entails the determination of exactly what it means in context. The parser correctly determines the meanings of fragments such as "throw in the towel," "throw the ball in the pit," "throw out the garbage," "throw out the court case," and "throw a party." The following complete sentences have been analyzed by WEP:

- (S.1) The deep philosopher throws in the towel.
- (S.2) The man throws a peach pit in the deep pit.
- (S.3) The judge throws out a case.
- (S.4) The man throws out a pit.

The current WEP easily analyzes sentences similar to these examples, and interprets them as having the particular meanings intended. In addition, a version of WEP at the University of Leuven has been shown (6) to analyze Dutch sentences with multiple lexical ambiguities particular to that language.

Word experts do not only model content words, however. In fact, they are not even restricted to words but also represent affixes, punctuation marks, sets of items, and nonlexical cues. The collection of implemented experts includes experts for function words "the" and "a," for suffixes "en," "s," and "ing," for the set of all integers, for the punctuation mark ".", and for capital letters. Unimplemented experts that have been studied include the expert for the relative "who," for unknown words, for the conjunctions "while" and "and," and for the dollar sign.

The word expert for the verb suffix "en" has two implemented usages, as the following analyzed sentences show:

- (S.5) The case was thrown out by the court.
- (S.6) The man has thrown a party.

In passive sentences, the "en" expert has a central responsibility to coordinate the analysis, as in sentence S.5, and in other contexts (e.g., sentence S.6) makes little contribution at all. The "ing" expert coordinates the different analyses that WEP performs for the following examples:

- (S.7) The man eating tiger growls.
- (S.8) The man eating spaghetti growls.
- (S.9) The man eating shrimp growls.
- (S.10) The spaghetti eating shrimp growls.

The "ing" expert is the one that figures out who eats whom (or what) in these cases. The final "." on all these sentences indicates a sentence break, and the "." expert is responsible for relating that fact to the model.

The word-expert parser analyzes fragments of text through a distributed word-based control structure. The organization of the system around the individual word-expert programs leads to interesting parsing behaviors. The parsing of passive sentences through the actions of expert programs modeling "en," "by," and "was," for example, suggests various things about parsing knowledge. One of the most provocative word experts for future research is the expert for unknown words. What does it mean for this expert to "fit its word into the overall interpretation" of some fragment? This entry suggests how WEP addresses such interesting topics, but the reading list should be consulted for more detailed articles.

### Formalizing Word-Expert Parsing

The representation of word experts has been the principal focus of attention during the course of the WEP model development. Recall the dual responsibility of each word expert to discriminate the intended role of its word in context and at the same time to provide useful information to its neighbors.

Word experts are represented as graphs composed of designated subgraphs that have no cycles, as shown symbolically in Figure 1. Each subnetwork consists of nodes that ask questions and perform side-effect actions. The focus of the WEP research has been the constant development, refinement, and redevelopment of these questions and actions, which comprise a taxonomy of knowledge for WEP. Although this epistemology does not yet satisfy strict adequacy criteria, it has nevertheless taken shape enough to be provocative (7).

The representation language for word experts makes up a formal theory of word-based parsing founded on the underlying notions of word-sense discrimination and active lexical interaction. The parsing process has been conceptually divided into these two fundamental aspects, and although the word-expert representation language comprises a single formalism, the natural separation between these two facets has been maintained. The word-expert formalism consists of the lexical-interaction language (LIL) and the sense-discrimination language (SDL). The syntax of the languages, presented formally in Ref. 7, specifies only the organization of word experts and not the semantics of the individual nodes comprising them. This section briefly describes their semantics, a more complete description of which appears in Refs. 7-9.

The word-expert representation languages describe the context-probing questions asked by word experts and the actions they perform to build concept structures and to interact with the other experts representing the words of the input sentence. The statements of the LIL include one causing the expert to AWAIT data from another expert, one to transmit a control SIGNAL to another expert, and the other to REPORT

```

<expert <name>
  [entry0
    (node0 <node type> <node body>)
    (node1 <node type> <node body>)
    ...
    (noden <node type> <node body>)]
  [entry1 ...]
  ...
  [entry_m ...]

```

Figure 1. Word-expert structure.

a conceptual entity to other experts and to the model as a whole. The statements of the SDL serve to build and refine concept representations, post expectations and constraints, and carry out similar operations aimed at determining the meaning and role of the particular word in the text at large. The questions and actions of word experts serve to analyze context and assemble structures (sense discrimination) and to communicate this information outside (active lexical interaction). These word-expert functions are summarized in Figure 2.

**Word-Expert Structure.** The "deep" expert shown in Figure 3 illustrates the basic structure of word experts and includes certain of the important statements of LIL and SDL. The body of a word expert consists of a collection of tree structures, each of which is called a continuation or entry point (entry) of the word expert. Each entry point is made up of action nodes and question nodes. An action node contains a number of structure-building or interaction statements with at most one successor node. A question node consists of a single context-probing question and a number of successor nodes corresponding to its possible answers. The individual entry points are connected together into a large network by an expert action causing a branch from one continuation to another and by expert resumption after suspension, which always takes place at an entry point. A word expert is a network of nodes organized modularly into tree structures called entry points, each a distinguished subgraph of the expert.

Each time a word expert gains execution control, four items are provided to it by the parser. One of these is the entry point where processing should continue. The first time the expert runs, it receives the designated starting point ENTRY0 as its incoming entry variable. Wake-up demons (qv) provide the continuation for subsequent reentries. The three other input messages to the expert are a concept structure, a control signal, and the expert sending them (i.e., its internal name). On

#### *Side-Effect Actions*

Ask that the next word be read and its expert initialized.  
Build a concept structure to represent some meaning or to use as a filter.  
Change the global state of the system (a description of the syntactic focus of the system).

#### *Multiple-Choice Questions*

Ask about the system state description.  
Ask about the word represented by a particular expert.  
Ask about the lexical origins of some concept structure.

#### *Lexical Interactions*

Report new information to the model at large or to a particular word expert making a request.  
Post a timed demon to await some piece of needed information.  
Suspend execution until some information arrives (or relevant demons time themselves out).  
Terminate execution after completing its diagnosis as to its role.

#### *Memory Interactions*

Request information from a discourse tracking process.  
Request an inference from a general semantic-memory mechanism.  
Request a constrained pattern-matching operation.

**Figure 2.** Word-expert functions.

```
[word-expert deep
  [entry0 (node0:q signal signal0
    [entity-construction node1]
    [* node2])
    (node1:a (declareg)
      (continue entry1))
    (node2:a) (openg entity-construction)
      (declareg)
      (pause entry1))]
  [entry2 (node0:q view concept1
    [=c#anything node4]
    [=c#person node1]
    [=c#artistic-object node2]
    [=c#volume node3])
    (node4:a (refinec concept1 =c#deep-entity)
      (report concept1))
    (node1:a (refinec concept1 =c#deep-intellectual)
      (report concept1))
    (node2:a (refinec concept1 =c#deep-art-object)
      (report concept1))
    (node3:a (refinec concept1 =c#deep-volume)
      (report concept1))]
  [entry1 (node0:a (createc concept2 entity)
    (refinec concept2 =c#1#deep)
    (refinec concept2 =c#deep-entity)
    (buildc concept3 entity
      (one of =c#person
        =c#artistic-object
        =c#volume
        =c#anything))
    (await concept entity
      (bindconcept concept1)
      (filter concept3)
      (report here)
      (wait group 1)
      (continue entry 2)
      (else entry3))))]
  [entry3 (node0:a (link concept2)
    (report concept2))]]
```

**Figure 3.** The deep expert.

the initial execution, the parser always provides the expert with a control signal, whether it is one sent explicitly by some other expert or the special signal posted as the current control state of the parser. Every expert's signal variable SIGNAL0 automatically refers to this initial incoming control signal. Note that the root node of the deep expert asks about the value of this signal.

The deep expert of Figure 3 contains four distinct entry points, the initial entry ENTRY0, a continuation from ENTRY0 called ENTRY1, and two restart entry points, ENTRY2 and ENTRY3. The AWAIT statement of ENTRY1 creates a restart demon to resume execution later at ENTRY2. If the awaited data does not arrive in time, however, the time-out condition of that demon resumes deep at ENTRY3. The entry points of word experts represent modular subexperts to perform well-contained functions for their overall sense-discrimination processing.

The two kinds of nodes within entry points ask questions and perform actions. Note in Figure 3 that each entry point contains several nodes, each node indicating its type as A (action) or Q (question). The root node of the deep expert asks a SIGNAL question, and the root node of ENTRY2 asks a VIEW question. All other nodes in the word expert perform side-

effect actions. The sole node of ENTRY1 performs five such actions, the BUILD action creates a new concept, REFINEC refines its conceptual value, the second BUILD constructs a concept for use as a filter and expectation, and the AWAIT mechanism creates and posts a restart demon. Any node having no explicit successor represents an expert suspension or termination node. If the expert has any outstanding restart demons, it suspends at such nodes; otherwise, it terminates. In the deep expert, for example, the node of ENTRY1 always suspends the expert because of the AWAIT contained there, while all action nodes of ENTRY2 cause expert termination.

**Model Organization.** The decentralized representation of parsing knowledge in word experts leads to an overall model organization to support exchange of information and distributed decision-making (i.e., agreement on overall meaning). The executing word expert is temporarily the coordinator of the entire parsing process, changing structures and invoking procedures in its environment. The control structure of the parser is a lexical-interaction control structure. The basic aspects of this distributed control for a particular word expert include suspending execution while waiting for needed information and resuming execution when that information becomes available. When word-sense discrimination requires some word expert to ask a question of another expert, it must wait to receive the answer before proceeding. When the information arrives, the word expert continues processing where it left off. Eventually, the word expert completes its sense discrimination and determines the intended meaning of its word in context.

The control flow that results from this coroutine control environment can be viewed as the movement of a window across the input stream. Inside the window control passes from word expert to word expert and back again, but the hearing of new words and the termination of old experts causes the overall processing window to move ahead. Figure 4 illustrates the control structure of the model in terms of this metaphor. The hearing of new words expands the right side of the window as the termination of word experts contracts the left side. Eventually, the window includes the last word of the input text, and the process terminates.

**Control Mechanisms: Expert Suspension and Resumption.** Any word expert that desires some piece of information through interaction with one of its neighbors specifies the nature of that information in a data structure called a restart demon. The demon has the responsibility of monitoring the data transmissions on the two central pathways in the model. When the requested datum appears, the restart demon provides it to its associated word expert, which continues where it left off. The execution of the statement does not cause the sus-

pension of the expert. The expert continues its processing as if nothing had happened. Of course, it cannot make use of the needed data until it arrives, and thus often it must choose to suspend its execution at the time of demon creation. Note, however, that the expert might simultaneously create several restart demons, all awaiting different information.

As a procedural distributed model, the word-expert parser does not contain some finite fixed-length buffer to represent a working memory but models memory limitation with processes that have a strictly limited life span. The human understander does not wait indefinitely for information that could aid his or her understanding, and the model cannot either. When a word expert posts a restart demon to await some piece of information from another expert, a companion time-out demon is initiated to monitor the duration of the wait. In the ideal case the desired information quickly becomes available in the model, and the awaiting expert promptly receives it. If the information does not appear within a certain time, however, the special time-out demon must command the awaiting process to go on without it. The time-out demon specifies both how long the expert is willing to wait for the information it desires and the name of an entry point within the awaiting expert where processing should continue if the associated restart demon "times out" instead. The units of measurement for time-outs are based on certain model events, including the number of syntactic groups created, the number of words read, the number of sentence breaks encountered, and the termination (as opposed to suspension) of particular word experts.

**Message and Memory Objects: Expert Interaction Data.** The restart demons and time-outs of WEP coordinate the interaction among cooperating word experts. This interaction involves sending and receiving messages of only two (well-defined) types: concept structures and control signals. Concept structures represent the meaning of fragments of text, referents, relations, conceptual expectations, and all other conceptual knowledge in the system and can be thought of as case frames, such as those of Ref. 10 (see also Grammar, case). Control signals represent syntactic and idiosyncratic lexical cues, query-reply handshaking among experts, cues to accompany concept structures and to indicate their significance, and similar kinds of stereotypic information. Controls signals are currently just unstructured atomic symbols. Every interaction in the parser involves the transmission of a single message, which includes one concept structure and one control signal.

**Example Analysis.** The WEP is a distributed system in which the bulk of knowledge centers on individual words of language, and thus the main memory requirements of the parser are not great. The word experts are recalled from mass storage as needed by the WEP interpreter and purged from core by the LISP garbage collector when no longer active. The parser requires about 120 kbyte of memory for its functions and about 40 kbyte for the control work space and active memory. With the interpreter, a total of about 256,000 LISP nodes are needed. A word expert of average size requires between 2000 and 4000 cells of random-access memory on file, with expert size directly related to the richness of contextual meaning of the represented word.

Word-expert parsing consists of determining the contextual meanings of those words making up each sentence to be interpreted and figuring out how they combine to form larger con-

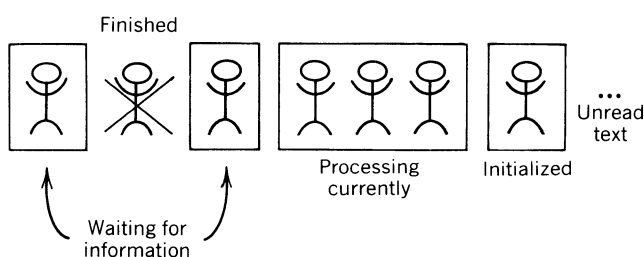


Figure 4. Word-expert parser control flow.

ceptual conglomerates. The process involves word-sense discrimination by the word experts representing the individual words of language carried out through the distributed control of active lexical interaction. Each word in a sentence contributes to the understanding of that sentence by some hearer in its own idiosyncratic way. Content words usually require significant processing of the local sense-discrimination variety, while function words and inflectional morphemes usually provide cues on how to combine meanings into larger structures. Those words playing syntactic roles are thus the control coordinators of distributed WEP.

**Example Sentence.** The following sentence has been chosen as our example because it combines in an interesting way the requirements for word-sense discrimination and active lexical interaction:

(S.11) The man eating peaches throws out a pit.

The analysis of the subject of this sentence requires complex word-expert interactions coordinated by the "ing" expert and knowledge of the real world to discriminate between a "man eating peaches" and the similar case of "man eating tigers." The "throw" expert must interact with the word expert for the particle "out" to determine its word sense. The man "throwing out a pit" must be distinguished from the man "throwing out a suggestion" or even "throwing out the window" something. The word expert for "pit" must decide, without benefit of a modification such as "peach" or "deep," whether it represents a fruit pit, a hole in the ground, a place for race cars to refuel, or some other concept.

The important aspects of a word-expert execution (note that each expert runs and suspends a number of times before terminating) consist of the input values to the expert, the entry points and nodes traversed, and the actions taken at each node. Since these elements cannot be fully elaborated for the present example due to space limitations, the reader is referred to Refs. 7, 8, and 11 for completely annotated program traces. Program traces for Dutch are in Ref. 6.

**Abbreviated Execution Trace.** Summarized below are all of the word-expert activations required to analyze the example sentence S.11. Note particularly the sequence in which the particular experts are activated and reactivated in order to process the input sentence.

*the.* The "the" expert initiates a new lexical group.

*man.* The "man" expert participates in the ongoing lexical group, reacting to the incoming signal from "the," and excludes the verbal meanings of "man." The "man" expert signals the end of the group and sends its concept representation back to "the."

*the.* The "the" expert had been awaiting the signal for the end of a group, so that it could report the concept to the system at large.

*ing.* The "ing" expert initiates a new lexical group and awaits a verb.

*eat.* The "eat" expert constructs a concept structure to represent "eating." Not enough information is available yet to refine the contextual meaning, so "eat" awaits it with a restart demon.

*ing.* The "ing" expert resumes and notes the name of its associated verb.

*s.* The "s" group (s1) initiates a new lexical group, which it knows from incoming context signals to be a noun group.

*ing.* The "ing" expert closes its lexical group and suspends again.

*s.* The "s" expert (s1) now prepares to await a signal indicating the end of a group and an accompanying concept structure.

*peach.* The "peach" expert receives a signal indicating the prior execution of "s" and thus rules out a modifier role in favor of a nominal role. Further, it begins the development of a concept structure to represent its meaning.

*s.* A new instantiation of the "s" expert (s2) opens a new lexical group, which it knows from incoming context signals to be a verb group.

*peach.* The "peach" expert ends the noun group and sends its concept structure to the awaiting "s" expert (s1).

*s.* The "s" expert (s1) grabs the concept structure and reports it to the model at large.

*ing.* The "ing" expert now resumes with the concept representing the meaning of the group to the right of "eating" in the sentence. Next it builds up two action concepts representing "man eats peaches" and "peaches eat man," determines that the former is plausible and the latter not, and sends the concepts to the model at large.

*eat.* The "eat" expert receives the concept representing its agent and refines its representation to reflect a person eating.

*eat.* The "eat" expert receives the concept representing its object.

*ing.* The "ing" expert resumes for the last time and cleans up.

*throw.* The "throw" expert receives a signal from "s" (s2) indicating a verb group and a concept representing its agent. Before determining its meaning, it must examine the word to its right.

*out.* The "out" expert begins executing with a signal indicating that a verb expert believes it to be a verb particle. The "out" expert awaits evidence either way.

*a.* The "a" expert begins a new lexical sequence.

*out.* The "out" expert posts a restart demon to await the upcoming nominal.

*pit.* The "pit" expert participates in the current group. It looks to its right for pairing words (e.g., "pit stop," "pit boss") and, not finding any, begins sense discrimination. This leads to the discourse expectations of something viewable as garbage, and a fruit pit fills the expectation.

*out.* The "out" expert resumes knowing that it pairs with "throw," and it sends "throw" a signal to that effect along with the object concept.

*throw.* The "throw" expert resumes with acceptance by "out" of its offer to pair up. Thus, "throw" completes its sense discrimination and terminates.

*\*per\*.* The "." expert forces certain restart demons to time out, e.g., the demons awaiting the temporal or spatial indices for the action. It terminates after sending a special sentence-break signal to the model.

**Working Memory after Analysis.** A number of concept structures are stored in the working memory of the model as a result of the analysis. These concepts represent the throwing away of garbage by a human adult male. The nature of the garbage is a fruit pit, and the man getting rid of it happens to be eating peaches. Although certain information deriving



from the meaning of the input sentence must still be inferred in higher-order mechanisms, the parser has made significant progress toward getting at the full contextual meaning of the sentence.

### Summary and Conclusions

The word-expert parser is a theory of natural-language understanding by computer and a program to implement partially that theory. The theory is based on the idea that the individual words of language are the primary sources of parsing knowledge and must be treated as such in a computer model. These word-expert knowledge sources coordinate the parsing process through active lexical interaction and the resulting patterns of behavior-model understanding.

Word-expert parsing has been motivated by observations about both human-language processing and existing computational efforts to engineer the process. People have an incredible facility for organizing and selecting word senses in arriving at the intended meaning of sentences in context. This takes place with such unnoticed and subconscious ease that it has been often overlooked. To those building language-parsing systems, however, this phenomenon represents a central problem that has been difficult to address. The reason for this lies in the inherent nature of the sense-selection problem, which can be viewed in terms of the rule-based problem-solving method (see Rule-based systems) as one of conflict resolution. Systems of rules capture similarities; they work best when there is a single choice at each decision point in the solution of a problem or when the process of choosing can be performed in a uniform way for all data. This situation is precisely opposite that involved in word-sense discrimination, in which the conflicts among applicable rules dominate the process. For accurately choosing the intended meaning of a word in context, the decision process must be tuned to the idiosyncrasies of that particular word. Uniform word-sense discrimination cannot succeed because each word interacts with its environment in particular knowledge-dependent ways to form meaning.

How can a language-understanding program account for all the meanings of individual words? The answer lies in focusing on the differences rather than the similarities of word usage in language. That means that instead of organizing the parsing system around rewrite rules, the system focuses on the individual word and on its idiosyncratic interaction with its context. The word-expert parser has done precisely that. Each word is represented as an active agent in its own right having the responsibility to fit itself into the overall meaning of the larger discourse structure in which it plays a part (see Discourse understanding).

The underlying framework for the organization and construction of word-based agents derives from discrimination networks. The nodes of a word-sense discrimination network comprise the context-probing questions of one word trying to determine its contextual usage. The arcs from the nodes represent the multiple choice of possible answers to the questions. The traversal of the network corresponds to the discrimination of the appropriate sense of the word in context. Each question probes one part of the context, and the answer chooses one path down the network, eliminating others. Eventually, the network traversal arrives at a terminal node, and the discrimination process has converged. Since the word-sense discrimination process is done on a word-by-word basis, the nature and

order of the context-probing questions are different for each individual word.

The development of a parsing system based on sense nets has depended on determining the nature of the questions they ask and devising methods for providing the answers. This question-answer process requires that the networks modeling individual words be able to communicate with each other to gain the contextual information that each one needs in determining its meaning. Besides interacting with each other, the word-expert questions must probe other higher-order aspects of context, such as the meaning of an overall discourse or the individual beliefs of the reader. Finally, each lexical agent must be able to postpone the answering of particular questions until enough information is available to do it correctly.

This entry has described WEP from two perspectives, that of the individual word, in which sense discrimination is primary, and that of the overall behavior of the distributed system, in which the pattern of lexical interactions is emphasized. The focus of research has been the representation of word experts and the development of a support environment for their processing. Word experts are represented with the LIL and the SDL. The statements of LIL permit an expert to ask questions of other experts and wait for the answers and to provide data to the system as a whole, often for use as answers to questions. The statements of SDL ask questions about discourse and real-world knowledge, build concept and signal structures, and coordinate internal network traversal. The patterns of behavior that arise during WEP are complex and provocative and suggest a new viewpoint on the nature of natural-language comprehension.

### BIBLIOGRAPHY

1. S. Johnson, *Dictionary of English*, W. Strahan, London, 1755.
2. D. Bolinger, *Aspects of Language*, Harcourt Brace Jovanovich, New York, 1975.
3. Y. Wilks, Preference Semantics, Technical Memo No. 206, Stanford Artificial Intelligence Laboratory, Palo Alto, CA, 1973.
4. C. K. Riesbeck, Computational Understanding: Analysis of Sentences and Context, Ph.D. Dissertation, Department of Computer Science, Stanford University, Palo Alto, CA, 1974.
5. M. P. Marcus, *A Theory of Syntactic Recognition for Natural Language*, MIT Press, Cambridge, MA, 1980.
6. G. Adrianes, Process Linguistics: The Theory and Practice of a Cognitive-Scientific Approach to Natural Language Understanding, Ph.D. Dissertation, University of Leuven, Leuven, Belgium, 1986.
7. S. L. Small, Word Expert Parsing: A Theory of Distributed Word-Based Natural Language Understanding, Ph.D. Dissertation, Department of Computer Science, University of Maryland, College Park, MD, 1980.
8. S. L. Small and C. J. Rieger, Parsing and Comprehending with Word Experts (A Theory and its Realization), in Lehnert and Ringle (eds.), *Strategies for Natural Language Processing*, Lawrence Erlbaum, Englewood Cliffs, NJ, 1982.
9. S. L. Small, A Word-Based Approach to Natural Language Understanding, in Bolc (ed.), *Natural Language Processing*, Springer-Verlag, Berlin, 1985.
10. C. J. Fillmore, The Case for Case, in Bach and Harms (eds.), *Universals in Linguistic Theory*, Holt, Rinehart and Winston, New York, 1968.

11. S. L. Small, Parsing as Cooperative Distributed Inference: Understanding through Memory Interactions, in King (ed.), *Parsing Natural Language*, Academic Press, New York, 1982.

### General References

- R. Berwick, "Transformational grammar and artificial intelligence: A contemporary view," *Cog. Brain Theor.* 6(4), 383-416 (1983).
- G. W. Cottrell and S. L. Small, "A connectionist scheme for word sense disambiguation," *Cog. Brain Theor.* 6(1), 89-120, (1983).
- J. A. Feldman and D. H. Ballard, "Connectionist models and their properties," *Cog. Sci.* 6(3), 205-254 (1982).
- M. P. Marcus, Wait and See Strategies for Parsing Natural Language, Working Paper No. 36, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1974.
- D. C. Marr, Artificial Intelligence—A Personal View, Technical Memo No. 355, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1976.
- D. McDermott and G. Sussman, The Conniver Reference Manual, Technical Memo No. 259a, MIT Artificial Intelligence Laboratory, Cambridge, MA, 1974.
- C. J. Rieger, Viewing Parsing as Word Sense Discrimination, in W. O. Dingwall (ed.), *A Survey of Linguistic Science*, Greylock, Stamford, CT, 1977.
- C. J. Rieger, The Importance of Multiple Choice, *Proceedings on Theoretical Issues in Natural Language Processing*, Urbana, IL, 1978: also TR-656, Computer Science Department, University of Maryland, College Park, MD, 1978.
- C. J. Rieger, "Spontaneous computation in cognitive models," *Cog. Sci.* 1(3), 315-354 (1977).
- C. J. Rieger and S. L. Small, "Toward a theory of distributed word expert natural language parsing," *IEEE Trans. Sys. Man Cybernet.* 11(1), 43-51 (1981).
- C. K. Riesbeck and R. C. Schank, Comprehension by Computer: Expectation-Based Analysis of Sentences in Context, Research Report No. 78, Department of Computer Science, Yale University, New Haven, CT, 1976.
- R. C. Schank, "Conceptual dependency: A theory of natural language understanding," *Cog. Psychol.* 3(4), 552-631 (1972).
- S. L. Small, Viewing Word Expert Parsing as Linguistic Theory, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC, 1981.
- S. L. Small, Demon Timeouts: Limiting the Life Spans of Spontaneous Computations in Cognitive Models, *Proceedings of the Cognitive Science Society*, Berkeley, CA, 1981.
- S. L. Small, G. W. Cottrell, and L. Shastri, Towards Connectionist Parsing, *Proceedings of the American Association for Artificial Intelligence*, Pittsburgh, PA, 1982.
- S. L. Small and M. Lucas, "A computer model of sentence comprehension," *Richerche di Psicologia* 25 (1983). Also appeared as Technical Report No. 1, Cognitive Science Program, University of Rochester, Rochester, NY, 1983.
- D. Waltz and J. Pollack, "Massively parallel parsing: A strongly interactive model of natural language interpretation," *Cog. Sci.* 9(1), 51-74 (1985).

S. L. SMALL  
University of Rochester

## PATH PLANNING AND OBSTACLE AVOIDANCE

Path planning and obstacle avoidance are catch words that imply a particular view of a more general family of problems. A better name for this family of problems would be "the plan-

ning and execution of actions." This entry presents a taxonomy of techniques for the planning and execution of actions, organized according to the temporal nature and certainty of the knowledge about the world that is available to the system.

The planning and execution of actions is the interface at which intelligent systems meets robotics (qv). As such, it is fundamental to the adaptation of AI technology to manufacturing and robotics. It is also an important problem domain for the science of AI. Because actions are so basic to human existence, the planning and execution of actions are problems about which humans can easily reason. Action planning has served as a paradigm for more abstract forms of planning (qv) and problem solving (qv). The two most famous examples are STRIPS (1) used for the SRI Shakey Project (see Shakey; Strips) and the blocks world (2,3). Going beyond simple static planning to dynamic plan modification and execution may well lead to other new insights into mechanisms for intelligence.

Mobile-robot actions (see Autonomous vehicles; Robots, mobile) are inherently simpler than actions of a robot arm; most mobile robots operate in a fundamentally two-dimensional (2-D) space and have two or three degrees of freedom (obvious exceptions are autonomous underwater vehicles and autonomous pilotless aircraft) whereas, robot arms operate in a three-dimensional (3-D) space and have five, six, or seven degrees of freedom. Perhaps more seriously, a planning system for a robot arm must also be concerned with the path of each of the links in the arm. Because mobile-robot actions are easier to visualize than the actions of a robot arm, many planning techniques have been first worked out considering a 2-D mobile robot and then extended to the case of a robot arm or to the more abstract domain of problem solving.

**Problem Definition.** There does not exist a single definition of the "path-planning problem" or the "obstacle-avoidance problem." Rather, there are an ensemble of techniques that assume varying degrees of knowledge about the external world. This family of problems might be summarized as the generation and execution of actions in order to move a device to a desired position and orientation without violating certain physical constraints.

The device could be a mobile robot, a robot arm, or any vehicle or mechanism that is to be controlled by an intelligent system. Although most of the techniques presented below can be applied to a large variety of devices, the techniques presented are illustrated using only the two most popular devices: a robotic manipulator (or arm) and a mobile robot. In much of this entry the word "device" is used to refer to the instrument to be controlled. This word should be understood to refer to either a mobile robot or a robotic manipulator.

For both mobile robots and robotic manipulators, the task is to move the device to a desired position and orientation without violating certain constraints. The most basic form of constraint is the requirement of avoiding collisions with other physical objects. More general constraints, such as avoiding certain areas whenever possible, may also be included with many of the techniques described below.

**Issue: Static Knowledge and Dynamic Knowledge.** A variety of techniques has been developed for planning and executing actions. Most of these techniques are based on different assumptions about the available knowledge of the world. These techniques range from pure rule-based planning techniques,



which assume a model of the world that is static, perfect, and difficult to modify, to road- and wall-following techniques, which dynamically process sensor data to produce action commands using feedback loops derived from classic control theory. Between these two extremes lie a variety of interesting techniques based on dynamically maintained models of the environment. Such techniques can function with a model of the world that is evolving and less than complete.

In most cases the system cannot be certain of geometric models of the limits of free space which are constructed in advance of plan execution. If the environment is not completely known in advance or is dynamically changing, such modeling necessarily involves sensing and dynamic model updating. Dynamic modeling is a topic covered by the field of 3-D vision systems. Dynamic modeling involves hard problems in 3-D sensing, surface or volume representation, and combining possibly conflicting information from multiple sensors and multiple views. The entries Vision, Stereo vision, Optical flow, Shape analysis, and Multisensor integration provide further information. The articles in Refs. 4 and 5 describe systems for incrementally constructing and maintaining a model of an environment. The article in Ref. 6 describes a 2-D dynamic modeling system for robot path planning using a rotating sonar. Moravec (7) describes a modeling system based on multiple-resolution stereo matching for robot path planning.

**Organization: Taxonomy of Techniques.** This entry uses world knowledge as the basis for a taxonomy of techniques for planning and execution of tasks. The first section will consider the ideal case where a model of the world exists in the form of a network of places. This network may be represented explicitly as a data structure or implicitly as a collection of rules in which the preconditions and postconditions are adjacent places. The first set of techniques do not tell how to generate the places in the network. An approach to generating a set of intermediate goals that leads to a minimum-length path is provided by the configuration space approach described in the second section. Minimizing the total length of a path may seem desirable, but such paths necessarily take the device close to obstacles where imprecision in the position of the device can cause collisions (the "too-near" problem). A safer path is given by keeping the device in the middle of the free space between obstacles. However, when obstacles are sparse, the middle of free space may take the device on an unnecessarily long path (the "too-far" problem). A compromise may be found by the use of techniques that mimic electrostatic fields. The taxonomy of techniques is rounded out with a discussion of "follower" techniques that track linear features such as roads or walls. The low-level control of such devices takes the form of a "pure-pursuit" algorithm, easily derived from techniques from classical control theory.

### Path Planning with Static Knowledge

The problem of planning and executing actions is generally organized as a hierarchy with at least three levels.

**Path planning** A path-planning system develops a plan to achieve a goal using an abstract model of the domain. In most cases the elements of this model have been prelearned (or constructed by hand) and cannot be directly sensed at the time of planning.

**Plan execution** A plan-execution system uses a dynamically maintained model of the local environment to monitor the execution of the plan and to execute actions. The execution system must verify that each step can be executed and watch for opportunities to simplify the plan.

**Action control** Low-level control of actions often involves techniques from control theory. In sophisticated systems such controllers are based on the dynamics of the device and on a dynamically maintained model of the structure of the environment.

Path planning involves search through a set of abstract descriptions of possible actions. In most systems such abstract descriptions have the form of a network of decision points.

**Basic Principle: Search over a Network of Decision Points.** Each uniquely identifiable position for a device can be thought of as a state. However, for any reasonable position resolution and any reasonable domain size, the number of possible position states is extremely large, overwhelming any technique for path planning. As in so many other domains, the method for coping with complexity is abstraction. For reasoning about paths in space, the continuum of position states is abstractly represented by a much smaller set of discrete positions. These positions should be organized such that for each position the device may reach a set of neighboring positions by some relatively simple, automatic, procedure. Furthermore, from any position it should be possible to eventually reach any other position. Thus, these positions are a set of states interconnected as a compact graph.

Travel between position states is carried out by lower levels of the system for monitoring the execution of the plan. For the path planner actions are atomic, and each action makes possible a set of other actions. Hence, position states are sometimes referred to as decision points, and the spatial arrangement of position states as a network of decision points. A good example of a network of decision points is a highway system. The intersections of the highway give decision points for driving. Between intersections, one has no choice except to follow the road—driving concerns avoiding other cars, construction workers, and careless animals (the plan-execution system), and with steering to follow the road (the action controller).

**Knowledge Representation.** Production-system architectures (see Rule-based systems) are very well suited for reasoning about things that can be expressed as a network of abstract states. Thus, a natural way to plan paths through a network of decision points is to represent each interconnection between a pair of decision points as a production rule. Paths may be planned by either forward chaining or backward chaining (see Processing, bottom-up and top-down) through the collection of rules, searching for the best path. A commonly cited simple example of such a system is the blocks-world program MOVER program (2) described in a textbook by Winston (8). A more sophisticated example is given by the STRIPS planning system (1) (see Shakey; Strips) used for path planning for the Shakey project at SRI (9). Path planning for Shakey served as a paradigm, or simple model, of the more general forms of problem solving.

In many examples of rule-based planning systems, the knowledge of the domain (see Domain knowledge) is encoded

in the production rules. Although knowledge encoded as production rules is very general, it can also be difficult to create and relatively slow and difficult to modify. The planning process can be made faster and more flexible by representing the domain knowledge as a network of decision points represented explicitly as a graph. The network of decision points may then be treated as a declarative knowledge base, from which knowledge of places may be copied to working memory for use with a rule-based search procedure. Alternatively, the network of places may be used directly by a procedural implementation of a graph-search algorithm (see Search).

**Path Planning with Heuristic (A\*) Search.** Nilsson (10) has provided a general form of search algorithm known as GraphSearch. When a heuristic cost function is available, the GraphSearch algorithm takes on a form known as the A search algorithm. A well-known result in planning theory is that if the heuristic cost function is always less than or equal to the true cost, the A search algorithm will give the optimum path to the goal (11,12). In this case the A search algorithm is known as the A\* algorithm (qv). Although this result is only of marginal value in many domains, it can be very useful for robot path planning.

In path planning, the true path cost is usually given by the network distance to the goal (the sum of the distances for each pair of decision points on the path). In some cases an additional cost is added for the number of decision points along the path. In either case the Euclidean distance to a goal will always be less than or equal to the network distance and hence provides exactly the heuristic needed to make the A\* search algorithm optimal (for details, see A\* algorithm).

**Exhaustive Search.** Heuristic search is only guaranteed to produce the least cost path when a cost estimate is used that is guaranteed to be lower than the true cost. If a more complex cost function is used, and a cost estimate does not exist that is guaranteed to be lower than the true cost, the GraphSearch algorithm takes on the form of a breadth-first search (see Search, depth-first). The exhaustive search algorithm with the lowest complexity that is guaranteed to find the least-cost path from a node in a graph to any other node in the graph, using an arbitrary cost function, is Dijkstra's algorithm (13).

In its complete form, Dijkstra's algorithm provides a solution to the "single-source shortest-path" problem. That is, the algorithm finds the least-cost path from a start node to every other node in the graph. The algorithm can be easily modified to halt when the least-cost path has been found for the desired node. It can be shown that this form of Dijkstra's algorithm is the least expensive algorithm that guarantees the least-cost path to the desired node using an arbitrary cost function.

**Generating the Decision Points.** In most cases the most difficult aspect of path planning using a network of decision points is actually generating the network of decision points. Techniques for generating the decision points generally follow one of two approaches: edge-following algorithms and free-space algorithms. Edge-following techniques can guarantee the set of decision points that lead to the shortest possible path through the intervening space. However, the shortest path may not necessarily be the most desirable path, as it generally involves passing adjacent to obstacles. Motion in the real world necessarily involves uncertainty in position, and as a device passes close to an obstacle, collisions become more

likely. One approach to minimizing the possibility of collision is to plan paths as far away as possible from objects. This is generally done using decision points based on a description of "free space." Another approach is to build the uncertainty in position into the path planning using "potential fields." All of these techniques are examined in the following sections.

### Configuration Space

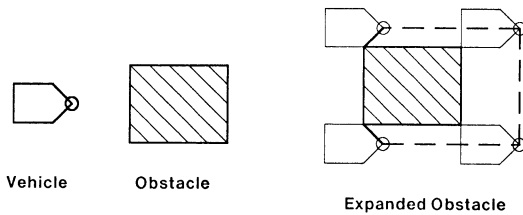
Unless the device is round, the orientation of the device is an important aspect of motion planning. If the device must maneuver through tightly spaced obstacles, the device orientation will be a crucial aspect of the plan. A classic form of the path-planning problem is the "piano-movers' problem"; that is, the problem of finding the right sequence of positions and orientations to move a piano through a sequence of obstacles (14). A modern form of this problem is the "find-path" problem, first formulated by Lozano-Perez (15,16). The object of the find-path problem is to find the sequence of positions and orientations that move an arbitrarily shaped device through a field of obstacles. Lozano-Perez proposed a technique known as configuration space for searching for an optimum sequence of positions and orientations. More recently Brooks (17) has proposed a variant of the configuration-space approach based on modeling the free space between obstacles.

**Configuration Space: The Basic Principle.** The basic idea of configuration space is to convert the problem to a space in which the vehicle may be thought of as a point. In configuration space obstacles are expanded by the shape of the device while the vehicle is simultaneously shrunk to the size of a point. Unless the device is round, an additional dimension is added to the space for the device's orientation. For a 2-D path-planning problem, this creates a 3-D space in which each plane corresponds to the vehicle being at a particular orientation.

**Expansion of Objects.** The expansion of an obstacle by the shape of the vehicle at a particular orientation can be defined mathematically as a function that labels a point in space as occupied or empty. A point  $(x, y)$  at orientation  $(a)$  in configuration space is defined to be occupied if the shape of the device intersects with the shape of an obstacle when the device is oriented at angle  $(a)$  with its center point at  $(x, y)$ . In theory, the choice of center point is arbitrary; the only criterion is that it must be consistent. In practice, the expansion algorithm can be simplified if one of the corners of the vehicle is used as the center point.

The expansion of one polygon by a second will also be a polygon, defined by a set of vertices. The problem is to find a set of such vehicle vertices that make up the vertices of the expanded obstacle. The borders of the expanded obstacle correspond to positions at which the vehicle is adjacent to the obstacle. Vertices in the expanded obstacle correspond to locations at which one or more vertices of the vehicle are just touching vertices of the obstacle. At such positions there is no overlap of the two shapes, only contact at vertices and along boundaries, as illustrated in Figure 1.

An efficient algorithm for finding the expanded obstacle can be based on placing the vehicle vertices at each obstacle vertex and determining if the vehicle area intersects the obstacle area. In this algorithm touching does not count as an intersection. For each position at which a vehicle vertex is placed at an obstacle vertex without creating an overlap between the vehicle and the obstacle, the vehicle position defines a vertex on



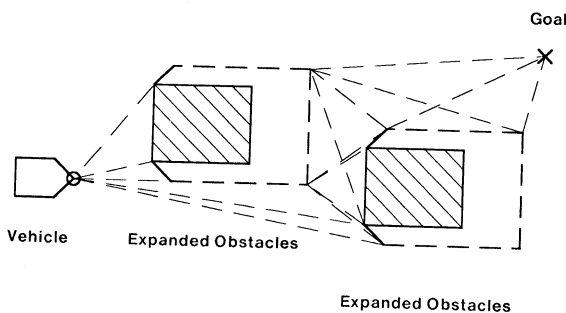
**Figure 1.** Obstacles are expanded by computing the polygon swept out by touching vehicle vertices to obstacle vertices.

the expanded obstacle. Both the vehicle and the obstacle must be represented by a convex polygon for this algorithm to work.

In configuration space the shortest path is either a straight line to the goal or a sequence of straight lines that pass through the corners of the expanded obstacles. The goal point, the start point, and the convex corners of expanded obstacles at each orientation plane provide points in a network of decision points, as shown in Figure 2. Planning a path becomes a problem of searching for the sequence of decision points that takes the vehicle from the start point to the goal point by the shortest path.

**Configuration Space for a Robot Arm.** The configuration-space approach is easily adapted to path planning for a robot arm. For a robot arm the dimensions in the configuration space are the angles of each of the joints of the arm (18). Combinations of angles at which any part of the arm would strike an obstacle define regions of this multidimensional space. As with the case of a mobile robot, if the shortest path is not a straight line between the start point and the goal, it involves straight lines that pass through the convex corners of one or more expanded obstacle regions. Thus, the convex corners of the obstacle regions again provide the decision points for path planning.

Robot arms typically have five or six axes, giving a space with a very high dimensionality. It is often possible to reduce the complexity of the configuration space by approximating the obstacles as separable over subdimensions in the space of joint angles. The forbidden regions in a separable approximation must include all points for which any configuration of the other joints would produce a collision. Projection into three sets of 2-D spaces or two sets of 3-D spaces will block entry to otherwise reachable regions of configuration space but will usually provide access to enough of the space to provide reasonable solutions. Paths planned in separable projections are not guaranteed to be optimum; they are guaranteed only to not produce a collision.



**Figure 2.** Corners in configuration space provide decision points that include the shortest path.

**Simplification: Representation as Spheres.** A central problem in obstacle avoidance is computing the intersection of shapes. An alternative to configuration space is to represent the vehicle or the objects with a representation that simplifies the calculation of intersections. One such representation is provided by describing the vehicle or objects as a hierarchical collection of spheres. Spheres are interesting as a basis for a representation because the intersection of a point with a sphere may be determined by comparing the distance of the point from the center of the sphere to the radius of the sphere.

The simplest case of a spherical representation is to describe the object or vehicle with the smallest sphere that encloses all of the object's volume. More accurate descriptions of the object may be organized as a hierarchy of descriptions with smaller spheres. An alternative to the expansion operation of configuration space is to represent the obstacles or the device as spheres or circles. This technique has been applied to the representation of obstacles for mobile-robot path planning (7) as well as the representation of a robot arm for collision detection (19).

**Application to Mobile Robot.** Moravec (7) developed a mobile-robot path-planning system based on a world model constructed from matching points in stereo images (see Stereo vision). Moravec used an "interest operator" that responds to high-contrast corners. When a pair of interest points are found to correspond in a pair of stereo images, it is possible to calculate an estimate of the depth of the physical point that corresponds to the corner. The correspondence of interest points from a stereo pair of images provides a point in 3-D space that is hypothesized to be occupied. Moravec's system determined 3-D points that were above the floor and plotted these as circles on a 2-D grid. Paths were planned on this 2-D grid as sequences of lines that passed tangent to circles. The intersection of potential paths with circles provided the decision points for path planning.

**Application to Robot Arm.** Myers (19) has demonstrated a collision-detection algorithm for a robot arm in which the links of the arm are represented by a hierarchical collection of spheres and rectangles. This system requires a description of the surfaces that might potentially be obstacles to the robot arm. Before executing a planned motion, the system chooses a sample set of arm configurations from the motion to test for intersection with the potential obstacles. For each configuration the system tests to see if any obstacle passes through large spheres enveloping the forearm. If there are no intersections with the large spheres, the system tests the next arm configuration. If an intersection is detected, the system tests for intersection with a more detailed description of the arm. If no intersection is detected for the more detailed description, the arm configuration is safe and the system passes on to the next arm configuration. If intersections are determined for the more detailed descriptions, the system notes the location of the intersection and uses this as the basis for computing an intermediate position that will avoid the intersection. The obstacle-avoidance algorithm is then computed for a path including the intermediate point.

### Free-Space Approaches

As mentioned above, the shortest path through a collection of obstacles is either a straight line or a collection of straight lines that connect the vertices of obstacles expanded by the shape of the vehicle. Because the shortest path involves pass-

ing adjacent to obstacles, any error in the vehicle's position can result in a collision with an obstacle. The likelihood of such a collision can be reduced by adding a tolerance to the shape of the vehicle. This tolerance represents the uncertainty in the vehicle's position. Fattening the vehicle by a position tolerance represents the uncertainty in vehicle position by a hard limit. In fact, such uncertainty may be more accurately represented by a Gaussian probability density function. Fattening by a hard limit to position uncertainty can have the effect of choking off access to doorways or spaces between obstacles through which the vehicle might normally pass. The route through a set of obstacles that minimizes the likelihood of collision passes through the center of the space between the obstacles. This observation has led to the development of path-planning algorithms based on representing the free space between obstacles as generalized cylinders and as convex regions.

**Generalized Cylinders.** Brooks has described path-planning systems for a mobile robot (17) and for a robot arm (18) based on representing the space between obstacles as 2-D "generalized cylinders." (qv) A 2-D generalized cylinder consists of a straight-line axis and sweeping function that defines the width of the cylinder as a function of position along the axis. The collection of cylinders that describe free space provide a set of "legal freeways" or "legal highways" over which the vehicle may travel. Decision points for path planning are provided by the locations where axes of cylinders cross. An example of a generalized cylinder description of a 2-D floor plan is shown in Figure 3.

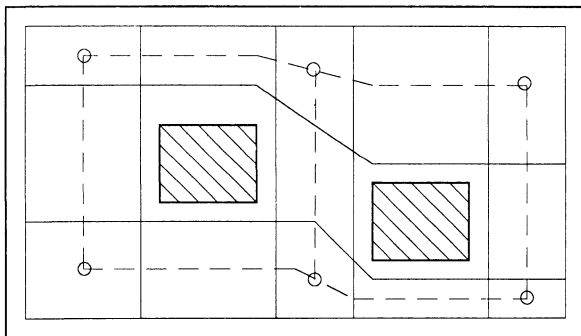
As noted above, the most general form of path planning for a robot arm involves paths in a space with one dimension for each axis of the arm. Brooks applied the generalized cylinder approach to path planning for an arm by representing the configuration space of the arm as a set of 2-D planes.

**Convex Decomposition.** A convex polygon is a region of a 2-D plane bounded by line segments such that the angle between each adjacent pair of lines is less than  $180^\circ$  (as measured from inside the polygon). For consistency between interior obstacles and outside walls, the vertex is referred to as "convex" if the angle measured in free space is greater than  $180^\circ$ . (Note that for the outside walls, this is the opposite of the notation used in geometry. For the case of obstacles in free space, this notation is consistent with that used in geometry.) Similarly, a convex polyhedral (or poly-tope) is a region of a

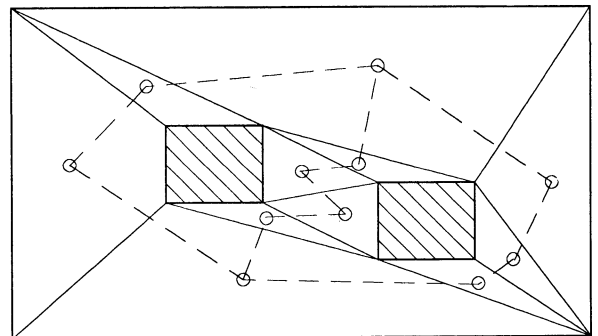
three- (or  $N$ -) dimensional space bounded by planar [or  $(N - 1)$ -dimensional] segments such that the interior angle between any two adjacent planar segments is less than  $180^\circ$ . Convex polygons, polyhedra, and poly-topes have the property that any interior point may be reached from any other interior point by a straight-line path that is entirely within the region. This property makes convex polygons (and polyhedra and poly-topes) useful representations for free space for use in path planning. Unfortunately, convex decompositions are not unique; for most polygons there are a variety of possible decompositions into convex regions.

**C-Cells.** Chatila (20) has defined a path-planning system that is based on dividing a prelearned floor plan into convex regions. Convex regions were formed by connecting nearest vertices to form areas called C-cells, as illustrated in Figure 4. Paths are then planned to pass through the center of C-cells. The system is used for the Hilare mobile robot developed at LAAS in Toulouse, France. Laumond (21) extended this idea by developing hierarchies of C-cells to represent rooms and parts of a known domain. These hierarchies were developed using graph-theory techniques to determine cut points in the graph of C-cells. A cut point is an arc of the graph that must be traversed to pass from one part of a graph to another. Subgraphs isolated by cutting at cut points are collapsed into a single point in a higher-level graph. Paths are then first planned using the higher-level graph and then elaborated using the subgraphs at the lower level.

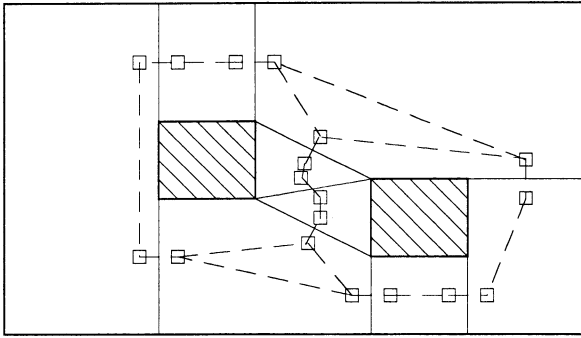
**Maximum-Area Decomposition.** Crowley (6) developed a navigation system in which a polygonal global model of the domain is divided into maximum-area convex regions. The maximum-area convex regions are defined by an algorithm that divides free space at each vertex that is convex with regard to free space. At each convex vertex, two segments are computed by projecting the segments that make up the vertex. A third segment is projected to the nearest visible vertex. The shortest of these three segments is selected as a partition of free space. The algorithm continues until no convex vertices remain. Each convex region is then shrunk by the sum of the radius of the robot and a tolerance factor. A pair of connected "adits" (entryways) are introduced at each segment that divides free space. Adits in each convex region are connected to all other adits within the region. The resulting network provides the network of decision points for global path planning. An example of the maximum-area convex decomposition of a floor plan and the resulting network of decision points is shown in Figure 5.



**Figure 3.** Free space may be represented as a network of 2-D generalized cylinders.



**Figure 4.** Connecting concave vertices to the nearest visible vertex partitions free space into convex regions.



**Figure 5.** Cuts that divide free space into maximum-area convex regions can provide the network of decision points for path planning.

### Potential Fields

One approach to compromise between the too-near problem of planning with corners in configuration space and the too-far problem of planning with free space has been the use of potential fields. The use of potential fields for obstacle avoidance has been independently developed by Khatib (22) and Krogh (23). A variation using path relaxation has been developed by Thorpe (24). These algorithms are similar to impedance-control techniques exploited by control theorists (25,26).

**Potential Fields.** In the simplest form of potential field algorithm, the model of the environment is sampled on a grid, and a repulsive function is computed at each point in the grid based on distance from the visible obstacles. The repulsive function is modeled as an electrostatic field, which gives a value at each point that is an exponential function of distance to visible objects. The field values are thresholded to provide a zero repulsion at a sufficient distance from all obstacles. The device is then considered as a point, and a path is planned by searching for the sequence of grid points that will take the device to the goal, while a cost function based on a weighted sum of distance and repulsion is minimized.

One approach to simplifying the cost of the potential field calculations is to compute the repulsive fields at points on a very sparse grid. Thorpe (24) has developed an algorithm that computes a path using a sparse grid. The path is then made "smooth" by a relaxation algorithm that moves the path onto a higher-resolution grid following the general form of the path on the original grid.

**Generalized Potential Fields.** Repulsive fields based only on distance can lead to paths that do not correspond to the device dynamics. This can lead to awkward path execution, and in the worst case the device is physically unable to follow the path. Krogh (23) has developed an algorithm for controlling the position and velocity of a device using a "generalized potential field." In this approach a vector field is constructed based on both the velocity and position of the device. At any point the field is computed as the sum of vectors generated by obstacles and attraction vectors generated by the goal. Obstacles are modeled as convex polygons to prevent local minima in the vector field, and the device is represented as a point.

Krogh's formulation of the generalized potential field includes velocity and the dynamics of the device in the calculation of the field. Although inclusion of the dynamics of the device provides smoother paths, it requires that the device

velocity be known at each point in order to calculate the repulsion vectors. This makes the generalized field at each point a 2-D function of the device velocity at that point. To calculate a global plan for a 2-D floor plan, the field must be calculated over a set of points in a 4-D space, which can quickly become very expensive. This approach can be more useful for obstacle avoidance in plan execution, provided that the controlling computer is capable of computing the repulsive fields with sufficient speed.

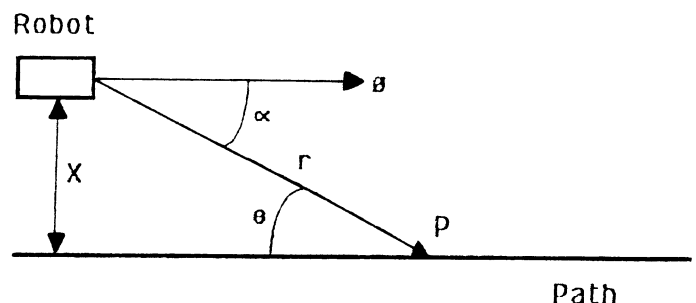
### Following

Global planning specifies the actions as a sequence of motions through decision points. During plan execution, the device position and velocity are controlled using information that is obtained from sensors. A particularly interesting class of low-level controllers are known as "followers." These devices use either raw measurements of the environment or measurements from structural descriptions of the environment to control the execution of planned actions. Followers are illustrated with examples of systems for following roads and walls.

**Road Following.** Wallace et al. (27) have developed a control algorithm that permits an autonomous mobile robot to use a vision system to follow a road. This system was implemented using the terragator robot developed at Carnegie-Mellon University. The terragator is a six-wheel-skid-steer vehicle with a motor controller that accepts independent commands for position and orientation and their first derivatives, i.e., forward speed and turning velocity.

An on-board camera captures images and transmits them via a microwave channel to a computer for processing. Experiments have been performed with a variety of techniques for rapidly extracting road edges from the images. Regardless of the image-processing algorithm, the output from the road-edge extractor is a pair of roughly parallel road edges in local robot coordinates. In this section a general form of "direct-pursuit" controller is described for following not only roads but also other liner features, stationary landmarks, and moving targets.

The general road-following problem is to find a steering function,  $F(P, \theta)$ , such that the robot will smoothly approach a specified path. The algorithm works by continuously controlling the orientation of the vehicle so that the vehicle points toward a point  $P$  on the center of desired path a distance  $r$  in front of the vehicle, as shown in Figure 6. To visualize the algorithm, imagine that a long stick of length  $r$  is rigidly glued to the nose of the vehicle, pointing straight ahead. The control system is designed to try to keep the end of the stick on the



**Figure 6.** Robot before turning toward path.

center of the road. This action directs the robot along a path that exponentially decays toward the center of the road.

Let  $\theta$  represent the current heading of the robot and let  $x$  represent the shortest (perpendicular) distance between the robot's current location and the path. Let  $\alpha$  represent the difference in orientation between the current heading and the heading toward  $P$ . Let  $\phi$  represent the difference between the heading that points from the robot toward  $P$  and the orientation of the path at point  $P$ . If the path is locally straight between the current position and the point  $r$ , it can be seen by simple geometry that

$$\frac{x}{r} = \sin \theta$$

If the robot travels at a velocity  $v$ , the perpendicular error  $x$  will be given by

$$\frac{dx}{dt} = -v \sin \theta$$

Substituting the first equation for the  $\sin \theta$  term in the second equation gives

$$\frac{dx}{dt} = -v \frac{x}{r}$$

This equation can be solved directly, giving

$$x = x_0 e^{-vt/r}$$

where  $x_0$  is the initial value of  $x$  at time  $t = 0$ . This equation shows that by continuously pointing the robot at the center of the road at distance  $r$ , the robot will exponentially approach the center of the road.

In fact, the robot cannot be turned instantaneously, and so the dynamics of turning must be included in the analysis. The displacement of the road centerline at a distance  $r$  is proportional to

$$\frac{\sin \alpha - \sin \phi}{\cos \alpha}$$

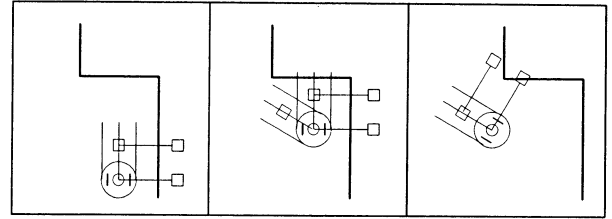
Servoing the rate of change of orientation,  $d\theta/dt$  on this quantity gives

$$\frac{d\theta}{dt} = -g \frac{\sin \alpha - \sin \phi}{\cos \alpha}$$

where  $g$  is the gain of the servo loop. This equation can be made linear by substituting  $Q = \sin \theta$ . This gives

$$\frac{dx}{dt} = -vQ \quad \text{and} \quad \frac{dQ}{dt} = -g \left( Q - \frac{x}{r} \right)$$

Solving these simultaneous equations yields three solutions, depending on whether  $g - 4v/r$  is less than, equal to, or greater than zero. All three cases lead to convergence to the centerline, but with different properties. When  $g > 4v/r$ , the control is underdamped, and the convergence is a decaying oscillation; the system overshoots the centerline. When  $g < 4v/r$ , the convergence is overdamped and approaches the centerline with a longer than necessary delay. In the case where  $g = 4v/r$ , the system converges to the centerline of the road with a minimum of delay. The turning rate of the robot, controlled by  $g$ , determines the behavior of the convergence. Note that  $g$  is dependent on both the robot's velocity  $v$  and the



**Figure 7.** Wall following with pseudosensor whiskers: left frame illustrates whiskers used in detecting presence of current segment and verifying next move. Boxes show ends of whiskers and next goal point. Center frame shows new segment detected by whiskers. Right frame shows device turning right in search of new segment to track.

distance  $r$ . To maintain a constant gain, the system must compensate for an increase in velocity by an increase in  $r$ .

Based on the above formula, a general-purpose control function of the form

$$\text{Follow}(P, \phi)$$

can be created. This function operates "asynchronously" to control the orientation of the vehicle and its first derivative; that is, each time the function is called, the current arguments replace the previous arguments with the new arguments. The cycle time for the function is typically determined by the speed with which the perception system can determine the location and the orientation of the desired path.

**Wall Following.** The equations for road following can be adapted to a large variety of devices and can be used to follow almost any linear structure that can be sensed and modeled. These equations can also be used to follow a moving vehicle or person. A simple form of wall following is reported by Crowley (6) where it is used for learning a global model of the environment. In this system a mobile robot uses a rotating sonar to construct and maintain a composite local model of the robot's environment. The robot senses and tracks a wall on its right side using a pseudosensor called a "whisker" that is computed using the composite local model, as shown in Figure 7. Side-looking whiskers are used to compute the distance to the wall at the robot's current location and at the location a distance  $r$  ahead along the current path. Other whiskers are projected ahead of the robot to detect barriers and concave corners. If either side-looking whisker does not intersect the wall, the robot reduces its speed and searches for the corner of the wall it was following. If a forward-looking whisker detects a new obstacle, the system begins tracking the new obstacle. When both side-looking whiskers detect the wall, a correction to the robot orientation is computed using the projected error at the forward whisker.

## Conclusion

This entry presents a survey of techniques for planning and executing paths both for mobile robots and for robot arms. Robotics is a particularly active research domain, and it is likely that even before this encyclopedia is printed, new approaches to these problems will be developed. Although it is difficult to predict the evolution of scientific research, some directions can be pointed out in which advances are likely. Two such directions are prominent, if for no other reason, be-



cause they are active areas of research that involve the extension of existing scientific paradigms.

The first likely area of new advances is in the area of perception and the integration of perception with plan execution and control of action. Historically, the heavy computational cost of perception has inhibited experiments in tightly integrating perception and action. As the cost of computing power decreases, systems that were previously unthinkable become possible and even commonplace. Two problems that have become particularly important are the integration of sensor data from multiple sources and the interface between a perception system and a system for action control. Reference 4 gives the author's current approach to these problems. Related ideas can be found in the blackboard architecture (see Blackboard systems) presented in Ref. 27. Knowledge of the structure of the environment is the basis for path planning and execution. Perception permits such knowledge to be constructed and maintained dynamically.

A second area ripe for new development is the control of action and perception with a production-system architecture. At the highest levels a mobile robot needs large amounts of both declarative and procedural knowledge. Procedural information, encoded as rules in a production system, can be a powerful and flexible representation for inference and problem-solving abilities. This is particularly true when problem-solving requires large amounts of poorly structured "how-to" knowledge. Such procedural knowledge is naturally complemented by declarative knowledge, primarily for cartographic information, such as a network of decision points. A declarative database is appropriate for information that changes frequently and with which a programmer may wish to interact graphically.

Currently, the author is experimenting with a production-system architecture based on OPS-5 (qv), in which cartographic information is held in a declarative database. Decision points are copied into working memory as needed, either during planning or during plan execution. Tasks such as road following and wall following are handled by algorithmic procedures that are invoked and managed by the execution of productions. Such systems will become increasingly important in the planning and execution of actions.

## BIBLIOGRAPHY

1. N. J. Nilsson, A Production System for Automatic Deduction, in *Machine Intelligence*, Vol. 9, Ellis Horwood, Chichester, UK, 1979.
2. T. J. Winograd, *Understanding Natural Language*, Academic Press, New York, 1972.
3. S. E. Fahlman, "A planning system for robot construction tasks," *Artifi. Intell.* 5, 1974.
4. J. L. Crowley, The Representation and Maintenance of a Composite Surface Model, *Proceedings of the 1986 IEEE Conference on Robotics and Automation*, April 1986.
5. M. Herman, T. Kanade, and S. Kuroe, "Incremental acquisition of a three dimensional scene model from images," *IEEE Trans. P.A.M.I.* PAMI 6(3), (May 1984).
6. J. L. Crowley, "Navigation for an intelligent mobile robot," *IEEE J. Robot. Automat.* 1(1), March 1985.
7. H. P. Moravec, Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover, Technical Report CMU-RI-TR-3, CMU Robotics Institute, Pittsburgh, PA, September 1980.
8. P. H. Winston, *Artificial Intelligence*, 2nd ed., Addison-Wesley, Reading, MA, 1984.
9. N. J. Nilsson, A Mobile Automation: An Application of Artificial Intelligence Techniques, *Proceedings of the First IJCAI*, Washington, DC, pp. 509-517, May 1969.
10. N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, 1980.
11. P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Sys. Sci. Cybernet.* SSC-4(2), 1968.
12. P. E. Hart, N. J. Nilsson, and B. Raphael, Correction to: "A formal basis for the heuristic determination of minimum cost paths," *SigArt Newsltt.* 37 (December 1972).
13. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Reading, MA, 1983.
14. S. Udupa, Collision Detection and Avoidance in Computer Controlled Manipulators, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, California Institute of Technology, Pasadena, CA, 1977.
15. T. Lozano-Perez, "Automatic planning of manipulator transfer movements," *IEEE Trans. Sys. Man Cybernet.* SMC-11, 681-698 (August 1981).
16. T. Lozano-Perez and M. A. Wesley, "An algorithm for planning collision free paths among polyhedral obstacles," *CACM* 22(10), 550-570 (October 1979).
17. R. A. Brooks, "Solving the find-path problem by good representations of freespace," *IEEE Trans. Sys. Man Cybernet.* SMC-13(3), (Mar/Apr 1983).
18. R. A. Brooks, Find-Path for a PUMA Class Robot, In *Proceedings of the Third National Conference of Artificial Intelligence*, Washington, DC, August 1983, pp. 381-386.
19. J. K. Myers and G. J. Agin, A Supervisory Collision Avoidance System for Robot Controllers, in *Robotics Research and Advanced Applications*, American Society of Mechanical Engineers, New York, 1982.
20. R. Chatila, Path Planning and Environmental Learning in a Mobile Robot System, in *ECAI*, Orsay, France, August 1982.
21. J. P. Laumond, Model Structuring and Concept Recognition: Two Aspects of Learning for a Mobile Robot, in *Proceedings of the Eighth IJCAI*, Karlsruhe, FRG, August 1983, pp. 839-841.
22. O. Khatib, Real Time Obstacle Avoidance for Manipulators and Mobile Robots, in *1985 IEEE Conference on Robotics and Automation*, March 1985.
23. B. H. Krogh, A Generalized Potential Field Approach to Obstacle Avoidance Control, in *Robotics Research: The Next Five Years and Beyond*, 1984.
24. C. E. Thorpe, Path Relaxation: Path Planning for a Mobile Robot, in *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, August 1984.
25. N. Hogan, Mechanical Impedance Control in Assistive Devices and Manipulators, in *Joint Automatic Control Conference*, August 1980.
26. J. R. Andrews, Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator. Master's Thesis, MIT, Department of Mechanical Engineering, Cambridge, MA, February 1983.
27. R. Wallace, A. Stentz, C. Thorpe, H. Moravec, W. Whittaker, and T. Kanade, First Results in Robot Road Following, in *Ninth International Joint Conference in Artificial Intelligence*, Los Angeles, CA, pp. 1089-1095. August 1985.

J. L. CROWLEY  
Laboratoire d'Informatique  
Fondamentale et d'Intelligence Artificielle



## PATTERN MATCHING

Pattern matching is a central problem in AI. Pattern-matching techniques are used in many areas such as natural-language understanding (qv), symbolic manipulation of algebraic expressions, expert systems (qv), program transformations, and synthesis.

Pattern matching is so basic to AI that it is provided as a primitive in many programming languages for AI. Most of the textbooks about LISP include one version of a pattern-matching program (1–3).

This entry starts by defining patterns and pattern-matching algorithms. It examines why and how pattern matching is used in AI. Examples of systems in which pattern matching plays an important role are illustrated.

The meaning of the word “pattern” in AI is not identical to the many meanings attached to that word in other areas of computer science. For instance, in pattern recognition (qv) a pattern is usually a feature vector in which each element of the vector represents the numeric value of a feature of the pattern being described. In syntactic pattern recognition a pattern is defined as a sentence generated by a grammar that describes the class to which the pattern belongs. In text processing a pattern is a string, and pattern matching is the operation that finds occurrences of a pattern within a text (see Natural-language generation; Natural-language understanding). Only the meaning of pattern and pattern matching used in AI is described here.

### Patterns and Pattern Matching

A pattern is a structural sketch of an item with some pieces left undefined. A pattern can be defined recursively as follows: a symbol is a pattern; a list of patterns is a pattern; and only expressions constructed by the rules given above are patterns.

The list notation allows the use of the same representation to describe any type of pattern independently of its meaning. This has the additional advantage that the same algorithms can be used independently of the content of the pattern.

A pattern can contain special symbols to indicate pattern variables. Variables are used in the matching process. A symbol that is not a variable is a pattern constant.

Pattern matching is the process of comparing patterns to see if they are similar. Two patterns are similar if they can be matched. A pattern matches another pattern if the two can be made identical by substituting some expressions for the variables in the patterns. A variable can match any expression except that every occurrence of the variable must be replaced by the same expression.

The matching process will fail if it is impossible to match the two patterns. It will succeed when the two patterns are identical or when they could be made identical with appropriate substitutions for the variables.

For example, assume that variable names are indicated by prefixing them by a question mark.

The pattern (THE ?X IS ?Y) can be matched with (THE ROAD IS WINDY) by replacing X with ROAD and Y with WINDY. The same pattern could be matched to (THE HOUSE IS (LARGE AND EMPTY)) by replacing X with HOUSE and Y with (LARGE AND EMPTY) but not to (JOHN IS THE PLAYER) or ((THE) TREE IS TALL).

In the previous examples ?X, ?Y are pattern variables. THE, ROAD, (LARGE AND EMPTY), etc., are pattern constants.

**Algorithm for Pattern Matching.** An algorithm for pattern matching follows. Assume that only one of the two patterns (called Pattern in the algorithm) contains variables. The second pattern (called Datum) does not contain any variable. The variables can be at any level of depth in the pattern. A simpler algorithm could be used if it is known that variables can appear only at the first level in the pattern.

There are three different types of results:

If the match fails, the result is a “NoMatch.”

If the match succeeds and the pattern does not have any variable, the result is “Empty.”

If the match succeeds and the pattern contains variables, the result is a list of bindings for the variables that will make the pattern equal to the expression. A binding is a pair of the form (variable expression).

When the algorithm is programmed in LISP, the results could be, respectively, the atom NIL, the list (NIL), and an association list containing the bindings. In this way it is possible to distinguish between the match failing, which returns the atom NIL, and the match succeeding with no bindings, which returns the list (NIL).

The description of the algorithm is in a Pascal-like notation. It assumes the existence of predicates that test patterns to see whether they are constants (IsConstant), or variables (IsVar). For example IsVar (?X) is True, IsConstant (THE) is True. It also assumes the existence of functions to perform manipulations on lists. The function FirstElementOf returns the first element of a list, RestOf returns a list except its first element, ListOf constructs a list out of its arguments, and Compose creates a list by appending its two arguments. VarSubst is a function that substitutes the binding of its first argument to all its occurrences in the second argument. Readers familiar with LISP will recognize that FirstOf is the LISP function CAR, RestOf is CDR, ListOf is LIST, and Compose is APPEND.

Match (Pattern, Datum):

{m1 and m2 are local variables}

if IsConstant (Pattern)

then if Pattern = Datum then return Empty

else return NoMatch

else

if IsVar (Pattern)

then return ListOf (Pattern, Datum)

else

if (m1 := Match (FirstElementOf (Pattern),

First ElementOf (Datum))) = NoMatch

then return NoMatch

else

```

if (m2 := Match (VarSubst (m1, RestOf (Pattern)),
    RestOf (Datum))) = NoMatch
then return NoMatch
else return Compose (m2, m1)

```

**Different Uses of Pattern Variables.** The variables used so far are called open variables. They can match any element. Once the variable has been bound, the binding is saved. The same variable can have only one binding.

Sometimes it is desirable to perform the matching only if the variable has already been bound. Those variables are called closed variables. A closed variable can match only the same element to which it has been bound. Closed variables are useful in interpreters of production rules when bindings have to be preserved between different calls to the pattern matcher (3).

Another important possibility is to match one variable with more than one element of a list, in general, with a sequence of any length. Those variables are called segment variables. They could be open segment variables or closed segment variables.

Another possibility is to have restricted variables. This means that there is a procedural restriction attached to the variable. The restriction can be any predicate. Only an expression that satisfies the restriction can match the variable.

A few examples follow. Assume that ?X and ?Y are open variables and +X, +Y, and +Z are open segment variables.

(F A ?X (?Y B)) can match (F A A ((G H) B)) with the bindings  $X = A$ ,  $Y = (G\ H)$ .

(F +X (+Y H)) can match (F A A (G H)) with the bindings  $X = A\ A$ ,  $Y = G$ .

(F +X ?Y +Z) can match (F A B C 2 W) with the bindings  $X = A\ B\ C$ ,  $Y = 2$ ,  $Z = W$  if the variable ?Y is restricted to match numbers only.

**Template Matching.** Pattern matching performed without explicit variables is usually called template matching (qv). Special symbols are used to indicate if the matching has to be done with a single entity or a sequence of them. Template matching is used in the case in which the bindings are not important. It has been used in early natural-language processing.

Assume that ? indicates an element that can match a single element, and + indicates an element that can match a sequence of elements in a list.

(F ? A) can match (F B A)

(F + A) can match (F B A) but also (F B B A) or (F B C H A)

**Unification.** If Pattern and Datum contain variables, the pattern-matching process is called "unification." It is more complex than the previous case because circular bindings should be avoided.

An algorithm for unification follows. OccursIn is a predicate that checks whether its first argument occurs in the second argument. This check is needed to avoid unifying an expression containing a given variable with that variable. For instance, if (F ?X ?X) is unified with (F (G ?X) ?X), the substi-

tution of (G ?X) for X cannot be accepted because substituting (G ?X) for ?X in the remainder of the expression would get into infinite recursion.

Unify (X, Y):

{m1 and m2 are local variables}

if X = Y then return Empty

else

if IsVar (X)

then if OccursIn (X, Y)

then return NoMatch

else return ListOf (X, Y)

else

if IsVar (Y)

then if OccursIn (Y, X)

then return NoMatch

else return ListOf (Y, X)

else

if (m1 := Unify (FirstElementOf (X),

FirstElementOf (Y))) = NoMatch

then return NoMatch

else

if (m2 := Unify (VarSubst (m1, RestOf (X)),

VarSubst (m1, RestOf (Y))) = NoMatch

then return NoMatch

else return Compose (m2, m1)

**Matching of Structures.** Sometimes it is more interesting to match structures instead of patterns. A structure can be, for instance, a set in which the order of the elements is irrelevant. Special data types and this kind of matching are provided in QLISP (4). Unfortunately, this feature is not available in other languages.

Another important area is the matching of part of graphs. This is used when knowledge is described in some form of network, such as semantic networks (qv) (5). It is not covered here because it is not considered pattern matching.

**Approximate Matching.** An interesting extension to conventional pattern matching is approximate matching, or matching when only a partial description is available (6). The program ELIZA (qv) (7) could be considered a pioneer in this respect. Little other work has been done in this area.

### Why and Where Pattern Matching Is Useful

There are two basic uses of pattern matching in AI systems, pattern-directed retrieval and pattern-directed invocation.

**Pattern-Directed Retrieval.** The basic idea is that information should be retrieved by pattern instead of by name. This

means that the name (or the computer address) of the stored information does not need to be remembered. Information is retrieved by content since the pattern describes the content of the piece of information to which it is attached. Even though pattern matching is basically a syntactic process, this allows one to keep in a better way the semantic meaning of pieces of data.

Implementation of pattern-directed retrieval has to be done with care. Scanning the entire database and testing each item against the pattern is computationally very expensive when the database is nontrivial. Various types of indexing mechanisms are used.

This technique is very important in expert systems. Consider, for example, a system based on production rules that does forward chaining (see Processing, bottom-up and top-down). A production rule is activated as a result of a pattern matching between the current content of the working memory and the antecedent part of the production rule.

In expert systems it has been found that often there is the need to match many patterns against many objects. The pattern matcher has to find every object that matches each pattern. This kind of pattern matching can be slow when large numbers of patterns or objects are involved. Methods to reduce the time needed in this case have been studied by Forgy (8).

**Pattern-Directed Invocation.** This method has been introduced with the design of the language PLANNER (qv) (9). The important idea is that procedures are called for what they do, not by their name. Each procedure has an associated pattern that in a sense describes what the procedure accomplishes. At decision points pattern matching is performed between the current situation or goal and the patterns of the available procedures. The procedure to execute is found in this way.

This makes control structures more data-driven since the procedures are invoked by the situation rather than being invoked in a preplanned sequence.

Since it is possible to have more than one procedure with the same pattern, and since it is possible that the matching process finds more than one pattern matching the goal, this method introduces nondeterminism. To handle nondeterminism, an arbitrary choice is made with the possibility to backtrack (see Backtracking) if later the choice appears to be incorrect.

### Examples of Systems Using Pattern Matching

Examples of systems in various areas of AI that use pattern matching are illustrated.

**Symbolic Manipulation of Algebraic Expressions.** Pattern matching is very important in systems for symbolic manipulation. One of the problems when doing symbolic manipulation is to simplify the resulting expressions. Pattern matching is excellent for this type of manipulation provided that simplification rules are expressed through patterns to be matched against the expressions.

SAINT (Symbolic Automatic INtegrator) was written by Slagle in 1961 (10). Its domain was elementary symbolic-integration problems. Pattern matching was used to transform the given expressions into equivalent expressions for which a closed solution was known.

MACSYMA (qv) (11) is a large system for symbolic manipulation of algebraic expressions. It has powerful pattern-matching capabilities that are very useful to simplify expressions. MACSYMA has simplification rules built in, such as simplify  $\sin(x + \frac{1}{2}\pi)$  with  $\cos x$ . The pattern matcher is used to find instances of the patterns in the simplification rules. The pattern matcher in MACSYMA is a semantic pattern matcher in the sense that it is able to find instances of patterns using nonsyntactic matches. For instance, the pattern  $ax^2 + bx + c$ , where  $a, b, c$  are pattern variables free of  $x$ , will match the patterns  $4x^2 + 4x + 1$ , or  $2x^2 + x - 1$ , or  $x^2$ . The most difficult problem with this type of pattern matcher is that it is difficult for the user to control the process of matching.

**Natural-Language Understanding.** Early natural-language understanding (qv) was done using a template matching approach. SIR (qv), STUDENT (qv), and ELIZA match the input against a series of predefined templates, binding the variables of the templates to the corresponding pieces of the input stream. This method has been abandoned in favor of more sophisticated methods based on a grammar for the language and a grammar parser. It is presented here because it works well when the topic of the conversation is limited; in that case the form of most of the input sentences could be anticipated and incorporated into templates.

SIR (12) accepted a limited subset of English as input. It matched the input against few templates of the form shown below.

\* is \*

\* is part of\*

Is \* \* ?

What is the \* of \* ?

The elements matched against \* were nouns possibly modified by a quantifier or a number. Even though the templates were very simple and limited, the system was able to answer a variety of questions.

STUDENT (13) was able to read and solve high-school-level algebra problems. Again the subset of English recognized was matched against a set of basic patterns of the type

(WHAT ARE \* AND \*)

(HOW MANY \*1 IS \*)

(\* (\*1/VERB) \* AS MANY \* AS \* (\*1/VERB) \* )

where \* indicates a string of words of any length, \*1 indicates one word, and (\*1/VERB) indicates that the matching element must be recognized as a verb in the dictionary. The simple pattern-matching schema combined with well-chosen heuristics did a very good job in parsing typical problems.

ELIZA (7) is the most famous pattern-matching program. Although the dialogue appears surprisingly realistic, the program works on pattern matching of input streams with templates. There are transformation rules associated with patterns. The input is scanned in search of a key word. Once one key word is found, the input is matched against patterns associated with the key word. The patterns are ordered so that the more complex are tried first. Each pattern is associated with some transformation rules that are used to generate the answer. For instance, (0 you 0 me) is a pattern in which 0 could

match any string of words. An associated transformation rule could be (what makes you think I 3 you) where 3 designates the third element matched in the original pattern.

**Problem Solving and Planning.** Pattern matching has been important in AI since the early days of GPS (General Problem Solver) (qv) (14). In GPS a problem is described by an initial state, a goal state, and a set of operators. An operator transforms one problem state into another. Each operator describes the conditions that have to be met for the operator to be applicable and the aspects of the problem state that will be changed by the application of the operator. A matching process is used to discover the differences between the goal state and the initial state. If there are differences, GPS tries to reduce them by applying an operator. This technique, known as means-ends analysis (qv) has been used in STRIPS (qv) (15) and in most planning systems (see Planning; Problem solving).

A simple planning system is part of SHRDLU (qv) (16). SHRDLU controls a simulated robot in a blocks world. The robot performs actions on blocks in answer to orders given in English. Even though the blocks world is used only as the domain of discourse, most of the basic concepts of planning have been incorporated in SHRDLU. The state of the world is described as a list of assertions. An assertion is a pattern. A goal is a pattern. An operator has a pattern that describes the effect of applying the operator and a body that consists of conditions to be checked or actions to be performed. The plan is constructed by successively matching goals or subgoals with patterns of operators until a sequence of actions that transforms the initial state into the goal state is found.

**Expert-System Shells and AI Languages.** Pattern matching is an essential component of any system based on production rules. Pattern matching is used to find the rules that match a goal (in backward chaining) or an assertion (in forward chaining). For instance, the pattern matcher in OPS5 is sophisticated and efficient. The R1 system (17), which is implemented in OPS5, does much pattern matching and almost no search.

KRL (6) has pattern-matching capabilities built into the language, even though it does not have any theorem-proving (qv) capability. Matches that require inference (qv) procedures can be performed only if rules for doing them are included as part of the knowledge.

PROLOG (see Logic programming) (18) is the best-known example of a language that contains a powerful unification algorithm and a theorem prover. Programs are written as a set of clauses. The arguments of predicates are unified by unification. The variable-binding process is used as a replacement of assignment statements. Due to the power of unification, the same program can be used to compute different types of results simply by exchanging the role of data and results. For instance, a PROLOG (qv) program that appends two lists X and Y into a list Z could be used to find the list Z obtained by appending the given list X to the given list Y. The same program could be used to find the value of the list Y that appended to the given list X produces the given list Z. Or, it could be used to find all the possible lists X and Y that when appended produce the given list Z. In the last case there are many bindings for X and Y that satisfy the given goal. In general, PROLOG returns only one binding per variable, but it could return all the possible bindings that satisfy a given goal.

**Program Transformation and Synthesis.** Pattern matching is often used in program transformation (19) to express locally applicable transformation rules. Those rules correspond to local refinements, such as

"1 + if B then x else y fi" is equivalent to "if B then 1 + x else 1 + y fi."

"L: if B then S; go to L; fi" is equivalent to "while B do S od."

Pattern matching has been used extensively in the system DEDALUS (20). It is interesting to observe that DEDALUS has been implemented in QLISP and that QLISP has perhaps the most complete pattern-matching capabilities of all the languages for AI.

**Intelligent Computer-Assisted Instruction.** Pattern matching has been used extensively for intelligent computer-assisted instruction (qv) (ICAI). Typically an ICAI system maintains a model of the student to be able to make hypothesis about his misconceptions and suggest corrections. The modeling of the student knowledge is often performed by simple pattern matching on the student's response history (21).

PROUST (22) uses pattern matching to match a student program with plans derived from the description of the desired behavior of the program. Even though PROUST has been applied only to simple programs, it shows again how the combination of pattern matching with problem-solving techniques produces important results.

## BIBLIOGRAPHY

1. H. Abelson and G. J. Sussman, *Structure and Interpretation of Computer Programs*, MIT Press, Cambridge, MA, 1984.
2. R. Wilensky, *LISPcraft*, Norton, New York, 1984.
3. P. H. Winston and B. K. P. Horn, *LISP*, 2nd ed., Addison-Wesley, Reading, MA, 1984.
4. R. Reboh, E. Sacerdoti, R. E. Fikes, D. Sagalowicz, R. J. Walslinger, and M. Wilber, QLISP: A Language for the Interactive Development of Complex Systems, Report N. TN-120, AI Center, SRI International, Menlo Park, CA, 1976.
5. N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, 1980.
6. D. G. Bobrow and T. Winograd, "An overview of KRL, a knowledge representation language," *Cog. Sci.* 1, 3-46 (1977).
7. J. Weizenbaum, "ELIZA. A computer program for the study of natural language communication between man and machine," *CACM* 9, 36-45 (1966).
8. C. L. Forgy, "Rete: A fast algorithm for the many patterns/many object pattern match problem," *Artif. Intell.* 19, 17-37 (1982).
9. C. Hewitt, Description and Theoretical Analysis (Using Schemata) of PLANNER, a Language for Proving Theorems and Manipulating Models in a Robot, Report N. TR-258, AI Laboratory, MIT, Cambridge, MA, 1972.
10. J. R. Slagle, A Heuristic Program that Solves Symbolic Integration Problems in Freshman Calculus, in E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, pp. 191-203, 1963. Also in *JACM* 10, 507-520 (1963).
11. MACSYMA group, MACSYMA Reference Manual, Technical Report, MIT, Cambridge, MA, 1977.
12. B. Raphael, SIR: A Computer Program for Semantic Information

- Retrieval, in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, pp. 33–145 (1968).
13. D. G. Bobrow, Natural Language Input for a Computer Problem Solving System, in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, pp. 146–226 (1968).
  14. G. Ernst and A. Newell, *GPS: A Case Study in Generality and Problem Solving*, Academic Press, New York, 1969.
  15. R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artif. Intell.* **2**, 189–208 (1971).
  16. T. Winograd, *Understanding Natural Language*, Academic Press, New York, 1972.
  17. J. McDermott, "R1: A rule-based configurer of computer systems," *Artif. Intell.* **19**, 39–88 (1982).
  18. R. Kowalsky, "Algorithm = logic + control," *CACM* **22**, 424–436 (1979).
  19. H. Partsch and R. Steinbruggen, "Program transformation systems," *Comput. Surv.* **15**, 199–236 (1983).
  20. Z. Manna and R. Waldinger, "Synthesis: Dreams  $\Rightarrow$  programs," *IEEE Trans. Sftwr. Eng.* **SE-5**, 294–328 (1979).
  21. J. A. Self, "Student models in computer-aided instruction," *Int. J. Man Mach. Stud.* **6**, 261–276 (1974).
  22. W. L. Johnson and E. Soloway, Intention-Based Diagnosis of Programming Errors, *Proceedings of the Fourth AAAI Conference*, Austin, TX, pp. 162–168, 1984.

J. SLAGLE AND M. GINI  
University of Minnesota

## PATTERN RECOGNITION

Automatic pattern recognition attempts to automate a class of perceptual and cognitive processes. These processes include extraction, identification, classification, and description of patterns in data gathered from real and simulated environments. Extraction is the task of processing raw data to derive intermediate results more representative of patterns of interest for the problem at hand. The design of a classifier involves analysis of sample data, modeling the variability among patterns belonging to the same class, determining the subset from an available set of measurements that adequately characterizes distinct categories, specification of techniques for extracting the measurement subset, and the design of the classification algorithm itself. At a more fundamental level the task of identifying whether or not patterns exhibit distinct characteristics for categorization and, if they do, which are those categories, are cognitive processes of concern in automatic pattern recognition. Complex patterns may be generated by specific interconnections of several primitive patterns; the generation of descriptions of these interconnections is the main problem in syntactic and structural pattern recognition. The understanding and exploitation of both the statistical and structural aspects of patterns have been pursued using techniques and ideas from many fields including formal linguistic theory and parsing (qv) methodology, geometry, physics, and other classical sciences and problem-solving (qv) methods of AI.

Pattern-recognition systems usually form parts of complex information-processing systems. A hypothetical advanced automation system is examined below to help understand how pattern-recognition problems arise in practice as well as how several practical problems may be formulated in this manner.

The scenario consists of a robot in a manufacturing plant. The robot looks around and moves in the environment. It receives parts from a conveyor, examines them for defects, determines the type of a part among those it expects to receive, decides where parts should be fitted, and assembles them. Occasionally, it takes instructions from a human supervisor and might also ask for help. The robot is a general-purpose machine in the sense that it can be assigned to one of several shop floors supervised by many people. The following are some of the pattern-recognition problems that need to be solved for the successful operation of these robots:

Training the robot to follow the speech of the assigned supervisor (limited-vocabulary speech-recognition (qv) problem).

Clustering (qv) to separate different types of parts using measurements on the sample or on its image.

Rejecting defective parts.

Generating descriptions of the assembly sequence.

Detecting situations requiring human intervention.

Generating descriptions to synthesize appropriate speech signals to ask for help.

Understanding speech instructions given by human operators.

A considerable amount of literature was generated on the application of statistical decision theory and discriminant analysis to the automatic classification of patterns. Subsequent criticism for focusing attention entirely on the statistical relationships among measurements and ignoring other structural properties led to proposals based on modeling patterns as sentences generated by a formal grammar. Today, in order to apply pattern-recognition techniques to solve problems such as those mentioned above, it is necessary to identify the fundamental mathematical problem that most closely resembles a practical problem at hand, make engineering approximations to the underlying functions, implement the scheme on an experimental basis, and test its performance before using it in a real-life situation. This entry presents a brief overview of the existing literature, techniques, and applications of pattern recognition.

Textbooks on pattern recognition cover either statistical or syntactic methods. Some of them cover both in distinct parts. Related books useful for pattern recognition are those on exploratory data analysis. Duda and Hart (1), Fukunaga (2), Tou and Gonzalez (3), Sklansky and Wassel (4), Devijver and Kittler (5), and Bow (6) are some of the textbooks on statistical pattern recognition. A popular way of organizing the material in statistical techniques is in the decreasing order of a priori information available to design recognition systems. Bayesian decision theory (qv) is directly applicable when complete information about a priori class probabilities, multivariate class conditional probability density functions over the feature (measurement) space, and the loss function of classification are available. At the next lower level of a priori information, the exact class conditional probability functions are not known; but their functional forms and a set of training samples from each class are available. The problem then boils down to the estimation of the parameters of the density functions. When it is not reasonable to assume any particular form for a density function, estimation of the entire density function is carried out by nonparametric methods. On the other

hand, it is possible to assume forms of the discriminant functions (as against density functions) for classification and determine the parameters of these functions from training samples. Linear discriminant functions are simple to design and implement. Finally, the lowest level of a priori information that can be made available for training a classifier is in a set of unlabeled training pattern samples. A common method of classifying them is by clustering. Anderberg (7) and Hartigan (8) are excellent books on clustering algorithms, their analyses, and their applications. An alternative to clustering is unsupervised learning (qv). This treats the problem as one of mixture density estimation. Whereas unsupervised learning trains a classifier, clustering yields a classification of the given set of unlabeled samples. The classified samples may then be used in a second stage to train a classifier. Bezdek (9) and Kandel (10) present fuzzy decision-making models as compatible but distinct companions to other statistical pattern-recognition models. Possible advantages claimed for fuzzy-set approaches are simpler decision functions, the ability to handle lack of multivariate statistics, and soft clustering with varying memberships of a pattern sample to all the categories.

Textbooks on syntactic pattern recognition include Fu (11), Pavlidis (12), and Gonzalez and Thomason (13). Syntactic pattern recognition views patterns as sentences or other higher level interconnections of primitives. The structure of interconnections decides the pattern category. Thus, techniques for syntactic recognition depend heavily on the theory of formal languages and the concatenation relationships. String and higher dimensional languages, syntax analysis, error-correcting parsing for strings, and error-correcting tree automata are treated in detail in Fu (11). Grammatical inference is the syntactic pattern-recognition analog of learning in statistical pattern recognizers.

Exploratory data analysis is a topic of importance to the successful design of pattern classifiers. Display of scatter plots, analysis of variance, factor analysis, and determination of linear transformations to lower dimensions retaining most of the variations in the data are crucial to feature selection, choice of a good technique for classification, and a preliminary assessment of the achievable performance. Many excellent books have been published on data analysis due to its wide applicability in engineering, sociology, psychology, and economics. Mosteller and Tukey (14), Wolff and Parsons (15), McNeil (16), and Chien (17) are some of the books on data analysis applicable to pattern recognition.

Research papers on the theory and applications of pattern recognition appear mainly in the following journals:

*IEEE Transactions on Pattern Analysis and Machine Intelligence,*  
*IEEE Transactions on Systems, Man and Cybernetics,*  
*IEEE Transactions on Information Theory,*  
*IEEE Transactions on Acoustics, Speech, and Signal Processing,*  
*IEEE Transactions on Geoscience and Remote Sensing,*  
*IEEE Transactions on Biomedical Engineering,*  
*Pattern Recognition,*  
*Pattern Recognition Letters,*  
*Journal of Classification,* and  
*Information Sciences.*

Important survey papers are Nagy (18), Ho and Agrawala (19), Kanal (20–22), Toussaint (23), and Fu and Rosenfeld (24). The International Joint Conference on Pattern Recognition and other conferences on pattern recognition and applications are held periodically. Recent important handbooks that include a wealth of information on pattern-recognition theory and applications include Krishnaiah and Kanal (25) and Young and Fu (26). Continuing developments are reported in a series of edited books entitled *Progress in Machine Intelligence* and *Pattern Recognition*. Books in the series that have already appeared include two volumes edited by Kanal and Rosenfeld (27), two by Gelsema and Kanal (28), one by Toussaint (29), and another by Rosenfeld (30).

### Models and Methodologies

Pattern recognition can be modeled by a variety of mathematical problems depending on the information available to the designer, the amount of training data, the required type of representation of patterns, and so forth. Figure 1 shows the highly iterative process of the development of a pattern-recognition system. Each task within the developmental process can be accomplished by well-established and/or new and innovative techniques. This section outlines many existing methodologies, their specific techniques, and the situations in which they are useful.

**Signal Processing.** Signal processing is the subject of transformations from one sequence of data to another to produce information in a form suitable for processing by pattern-recognition techniques. Such transformations are very useful for a variety of reasons including dimensionality reduction. Signals such as speech, sonar, and textured images, which are required to be classified into a number of categories, are inherently high dimensional. A straightaway modeling of the inter- and intraclass variability falls for several reasons, including the effect of insufficient training samples. The signal-processing approach represent these signals as functions of a few parameters and a long sequence of random numbers. The random numbers are from the same distribution for all the pattern classes. Therefore, distinct classes are modeled by the relatively small set of remaining parameters. For classification, we first pass the pattern through the signal processor to estimate the parameters. Examples of other uses of signal processing in pattern recognition are noise removal by filtering and curve fitting for detection of primitives in syntactic pattern recognition. Srinath and Rajasekharan (31) is a useful book for pattern-recognition applications of signal processing.

**Decision Theoretic Model.** One of the earliest pattern-recognition models used statistical decision theory. In this model each pattern is represented by a vector of measurements  $u = [u_1, \dots, u_N]^t$  (also known as features, observations, attributes). The statistical distributions of these features given that a pattern belongs to a known category  $\omega_i \in \{\omega_1, \dots, \omega_M\}$  is a known function  $p(u|\omega_i)$ . The decision theoretic problem is given  $u$ , decide the category,  $\omega_i$ . Many approaches are possible depending on the criterion of optimality. Bayesian decision theory assumes that the categories occur with a priori probabilities  $P(\omega_i)$  and decides the category with the maximum a

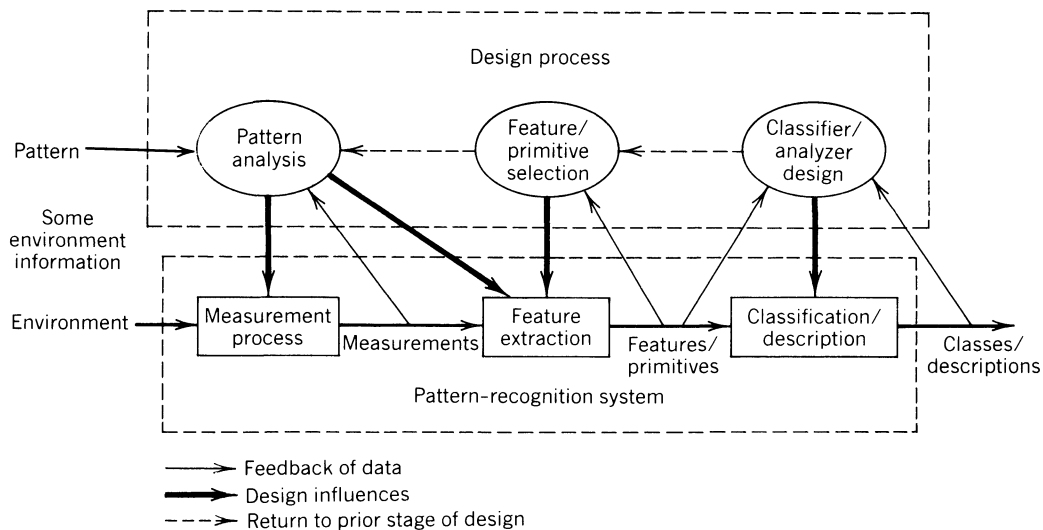


Figure 1. Iterative development of a pattern-recognition system.

posteriori probability. Thus, for an observation  $u$ , the class decision is  $\omega_k$  such that

$$P(\omega_k|u) = \max_{i \in \{1, \dots, M\}} \frac{p(u|\omega_i)P(\omega_i)}{\sum_{l=1}^M p(u|\omega_l)P(\omega_l)} \quad (1)$$

The performance of such a classifier is represented by the probability of correct classification given by

$$P_c = \int_u \left\{ \max_{i \in \{1, \dots, M\}} p(u|\omega_i)P(\omega_i) \right\} du \quad (2)$$

In many applications, different combinations of classifications and misclassifications are associated with different quantities of loss. Let a pattern belonging to class  $\omega_i$  incur a cost  $\lambda_{ij}$  if it is classified as  $\omega_j$ . The above Bayesian theory is extended to minimum-risk classification by minimizing the expected loss conditioned on observations:

$$\min_{j \in \{1, \dots, M\}} \sum_{i=1}^M \lambda_{ij} \frac{p(u|\omega_i)P(\omega_i)}{\sum_{l=1}^M p(u|\omega_l)P(\omega_l)} \quad (3)$$

The problem is complicated if the a priori probabilities are not known. A simple solution is to assume that all the classes are equally likely. The resulting maximum a posteriori probability decision is known as the maximum-likelihood decision. A conservative approach is to express the minimum risk as a function of variable a priori probabilities and then maximize the minimum risk with respect to the a priori probabilities. The resulting classifier is known as the minimax classifier. Many two-class pattern-recognition problems are detection of a hazardous event (warranting an alarm) against a null event in the presence of noise. Often, the user specifies a tolerable false-alarm rate. The scheme of minimizing the rate of miss constrained to a specified false-alarm rate is called the Neyman-Pearson scheme.

The above techniques devise decision mechanisms under the assumption that the class conditional distributions of the features are available. When such information is not available, the distribution functions need to be estimated from

training samples. Estimation of characteristics of pattern classes is an essential part of the design of decision theoretic pattern classifiers. In supervised estimation sets of patterns, each set drawn from one class, are available. If the functional forms of underlying class conditional distributions are known or can be assumed, the problem reduces to the estimation of a finite number of parameters. Gaussian distribution forms are very popular for many reasons. An  $N$ -dimensional Gaussian distribution is completely represented by an  $N$ -dimensional mean vector and an  $N \times N$  covariance matrix. Estimation of these quantities is simple. The distribution functions are smooth and are defined over the entire  $N$ -dimensional vector space. Resulting decisions turn out to be at most quadratic functions of the observation vector  $u$ . The central-limit theorem in the theory of random variables justifies Gaussian approximations when a random variable is a result of a combination of several physically independent effects.

Techniques for parameter estimation fall into three broad categories: point, interval, and Bayesian estimation. In point estimation the parameters are taken to be unknown constants. An explicit function of the observation is used as the estimate of the parameter. Maximum-likelihood estimation is one such technique. This substitutes a dummy variable (or dummy variables in the case of a vector parameter) for the unknown parameter and expresses the probability of observing the given sample set as a function of the dummy variable. The value of the dummy variable at which this probability is maximum is the maximum-likelihood estimate. In interval estimation the interest is in computing an interval that contains the parameter (being estimated) with a specified probability (confidence level). Bayesian estimation is appropriate if we can assume that the parameter is a random variable with an a priori density. The density is updated as more samples are given for estimation. Updated densities get sharper with the number of samples. The value of the parameter is taken to be the one at which the updated density peaks. A class of density functions known as the exponential family have the reproducing property in that the a priori and the updated densities are all of the same functional form, differing only in the values of some parameters. Another desirable property for both point and Bayesian estimation is the existence of a low-



dimensional sufficient statistic. The sufficient statistic (if available) allows us to represent a large set of independent samples from the same distribution by a small number of numbers from which the estimate can be computed. This is very helpful in recursive estimation

**Nonparametric Techniques.** Nonparametric techniques in pattern recognition are concerned with density estimation and classifier design using a set of labeled samples when there is insufficient information to assume forms of class conditional density functions. Nonparametric density estimation deals with pattern samples from one class and attempts to fit a density function spanning the entire feature space. In the Parzen window and the potential function method, the value of the density function at a point in the feature space is expressed as the cumulative contribution from individual samples used for estimation. A sample contributes to the density function through a window or a density kernel function. Examples of these kernel functions are rectangular functions and truncated squared-cosine functions. The potential function method treats a contributing sample as an electrically charged particle; the contribution to the density function is proportional to the potential at the point of interest due to the contributing sample. The  $k$ -nearest-neighbor density-estimation procedure computes the value of the density function at a point by a fraction that depends inversely on the volume of the region in the feature space enclosing the  $k$  nearest neighbors of the point in question.

Whereas nonparametric density estimation is one approach to overcome the lack of knowledge of functional forms for class conditional densities, the other approach is to directly design a classifier from labeled design samples, bypassing issues of individual density functions. Chief among such classifiers are the  $k$ -nearest-neighbor, linear and piecewise linear classifiers. The nearest-neighbor classifier assigns that category to a test pattern sample that is the class label of the geometrically closest design sample (a test sample is one whose category is to be decided by classification; a design sample is one from the set of labeled samples used to train a classifier). The technique is easily extended to  $k$  nearest neighbors. An interesting property of the nearest-neighbor classifier is that as the number of design samples tends to infinity, the error rate of the classifier is bounded by twice the Bayesian error rate (Bayesian error is the minimum achievable error). Linear classifiers designed from labeled samples are popular due to their simplicity of operation and the availability of optimization techniques applicable in their design. In the two-class problem, when the design samples are known to be separable into respective classes by a hyperplane, design techniques include the perceptron learning algorithm, the gradient descent, and the Ho-Kashyap (1) procedures. The general approach is to solve for a set of linear inequalities if the design set is linearly separable or to minimize the mean square distance of misclassified design samples from the discriminating plane if they are not.

**Unsupervised Methods.** Unsupervised methods are useful when the design data consist of unlabeled samples. It is then necessary to determine whether or not patterns exhibit distinct characteristics in order to categorize them. In unsupervised classification the given sample set is clustered into a number of groups. An important criterion for clustering is the minimization of the sum of squared error from the cluster means. If  $c$  is the number of clusters,  $u_{ij}$  the  $j$ th sample in the

$i$ th cluster, and  $\mu_i$  the centroid of the  $i$ th cluster, the sum of squared error is given by

$$J_e = \sum_{i=1}^c \sum_j |u_{ij} - \mu_i|^2 \quad (4)$$

Therefore, given a set of samples  $u_k$  and the number of clusters  $c$ , the clustering problem is to group them into  $c$  sets such that  $J_e$  is a minimum. A related criterion for minimization is the ratio of the sum of the intracluster squared distances to the sum of the intercluster squared distances. The sum of intracluster squared distance is the sum of the squared distances of each sample from the centroids of the respective clusters. The sum of the intercluster squared distances is the sum of the squared distances from cluster centroids to the mean of all the cluster centroids. A variety of other criteria using scatter matrices is also used. All these criteria lead to the same final cluster configurations provided the samples are well separated. A set of samples forming  $c$  clusters is well separated if the distance between any pair of samples within any cluster is less than the distance between any sample in one cluster to any sample in any other cluster. Algorithms for clustering based on the above criteria usually use iterative optimization techniques that guarantee only a local minimum starting from an initial partition of the sample set. Some of the other clustering methodologies are the hierarchical clustering schemes that successively merge nearest pair of distinct clusters starting from as many clusters as the number of samples, graph-theoretic clustering, and the  $k$ -means procedure. See Anderberg (7) and Hartigan (8) for details. Unsupervised learning differs from clustering in that unsupervised learning schemes attempt to learn the characteristics of the individual categories from unlabeled samples for use in training a classifier. The standard formulation is the estimation of the parameters of a mixture density. However, analytical solutions do not exist even for very simple forms of densities, and numerical solutions require nonlinear programming.

**Multistage Classification.** Any mathematical transformation from the feature space  $\{u\}$  to the pattern category space  $\{\omega_1, \dots, \omega_M\}$  is a pattern classifier. The distinction between types of classifiers rests in the way the transformation is designed and/or implemented. All the classification techniques described above are the so called one-shot methods. In contrast, multistage methods take several partial decisions in a sequence. Each decision can be viewed as a transformation from a subfeature space to an intermediate decision set (the intermediate decision set is not necessarily a subset of classes). The transformation at a stage depends on the previous decision. The resulting phased decision structure is often referred to as a decision tree. There are many advantages in using multistage decision-making structures: In many instances design of partial decisions is conceptually simpler as they are required to examine the information relevant to the present stage only. This would also save the expense of gathering information not required for the sequence of decisions a pattern sample may encounter. The flexibility of the multistage decision-making model enhances its applicability in a wide range of problems. However, this less restrictive representation opens up so many possibilities in tailoring a multistage scheme that there is no straightforward method to design a decision tree. Theoretically, when the a priori class probabilities, class conditional feature distributions, and misclassification

tion cost function are known, the optimal pattern classifier turns out to be a decision tree only if feature measurements have costs associated with them [see Dattatreya and Sarma (32) for details]. In practice, the optimal tree can be computed only for discrete features with moderate dimensionality and a small number of outcomes per feature. Figure 2 shows such a decision tree designed for spoken vowel recognition after a feature reduction transformation and discretization of the resulting eight normalized autocorrelation coefficients into three levels each. The three branches emanating from every intermediate node are the three decisions for low, medium, and high autocorrelation coefficients in the left-to-right order.

Other methods for decision tree design usually define frameworks for classification schemes to take advantage of the peculiarities in the data. Independent subrecognition systems design several classification systems, each to distinguish one class from the rest. A combination of decisions at the second stage yields the final decision. Such schemes are useful when it is possible to identify different feature subsets each of which highlights one class. Hierarchical classifiers reject classes in successive stages until one class remains. They are useful if groups of classes form a hierarchy and different groups can be distinguished by different features and/or decision functions. Dynamic tree-development methods are usually data analytic, and they successively split features to separate data sets belonging to different classes. A detailed review of decision-tree approaches to pattern recognition appears in Dattatreya and Kanal (33).

**State-Space Models.** In many practical problems it is possible to arrive at a partial structure of multistage classifiers. The class hierarchy and the features best suited for discrimi-

nation among the subgroups of classes may be available from a priori problem knowledge or data analysis. But such a structure is not a complete classifier since no decision rules are available. Concepts and search procedures from AI help to work out decision strategies. These strategies view classification as a search for a goal node in a state-space graph in which terminal nodes are class labels and other nodes specify features to be measured. The costs associated with the classification process are the measurement cost at every measurement node and classification risk at goal nodes. Two search strategies have been used. The *S*-admissible search strategy minimizes the total cost up to and inclusive of the decided goal node when the risk of classification is not influenced by measurements at nodes not along the path to the optimal goal node. If a good state-space graph is worked out in an interactive design phase, this property would hold approximately. If not, the *B*-admissible strategy allows for risk to be dependent on all measurements. Further details can be found in Kulkarni and Kanal (34).

**Syntactic Pattern Recognition.** The structural information about the interconnections in a complex pattern cannot be handled very well by statistical pattern recognition. Since a large number of distinct interconnections occur in practical pattern environments, it was found necessary to devise techniques to describe a large number of similar structures by the same category while allowing distinct descriptions among categorically different patterns. As an example, consider a single pictorial pattern in Figure 3. The accompanying description is shown in Figure 4. A slight movement in the objects would not change the description, but an interchange of the positions of the two objects would. Notice the similarity of the description

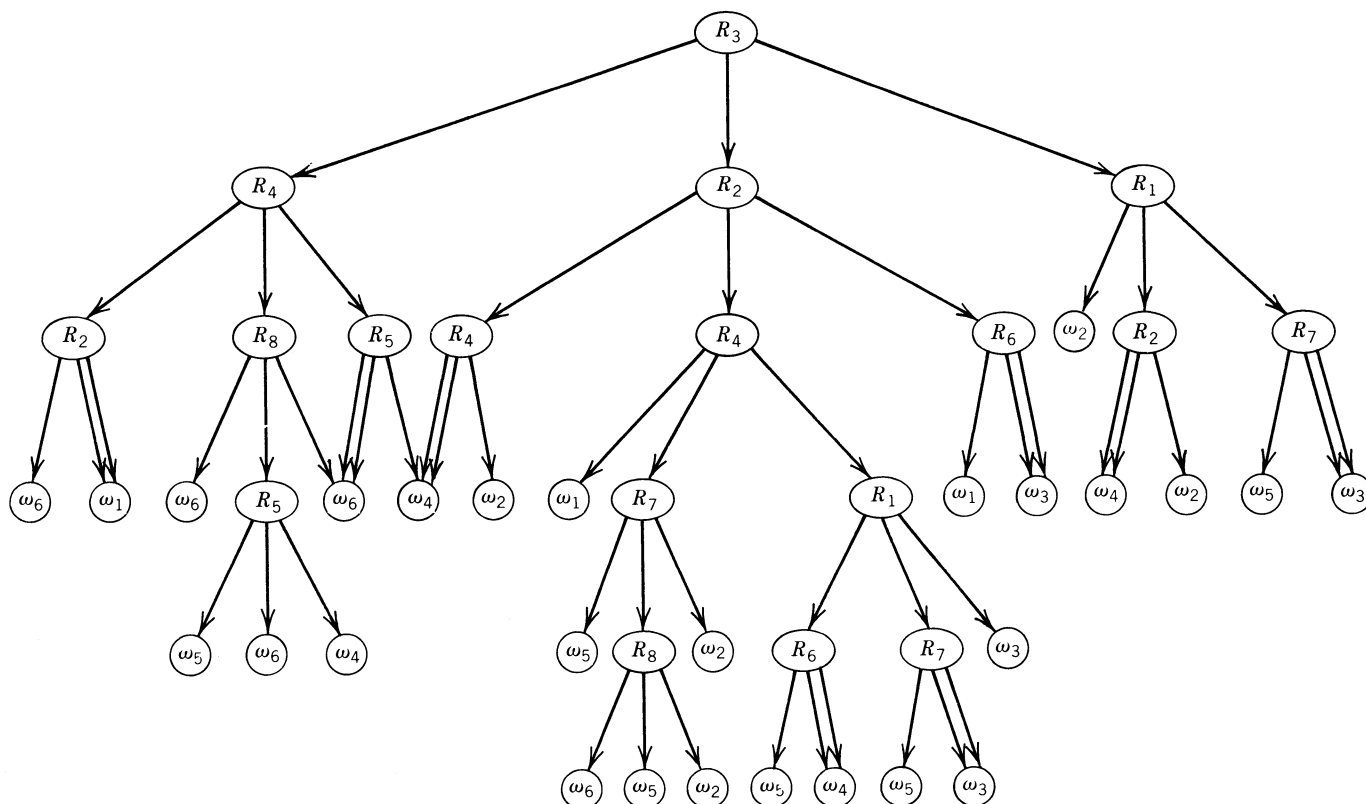


Figure 2. Decision tree designed for spoken-vowel recognition.

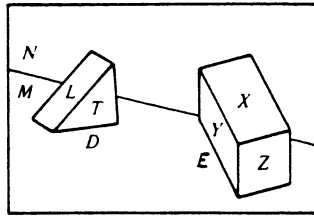


Figure 3. Pictorial pattern.

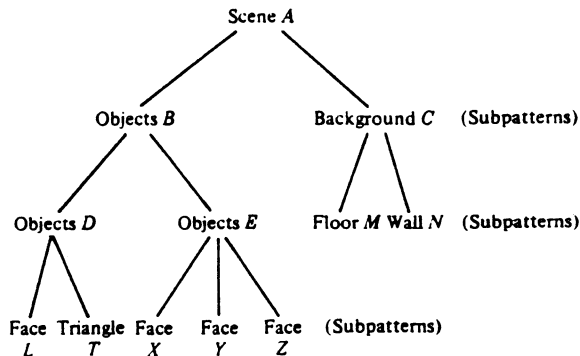


Figure 4. Description of pictorial pattern of Figure 3.

to the syntactic parsing of sentences in a language. Figure 5 shows the development of a syntactic pattern-recognition system. The analog of the conventional feature extractor is the primitive detector in syntactic pattern recognition; that of the classifier is the syntax analyzer. The training or the learning procedure in statistical pattern recognition corresponds to grammatical or structural inference in syntactic pattern recognition. Segmentation, decomposition, and detection of primitives in a syntactic pattern are usually carried out by signal processing, curve fitting, or other conventional techniques. The type of primitives used for representing pattern samples varies with application areas. The primitives should serve as basic building blocks to provide a compact but adequate description of the patterns in terms of the specified structural relations. They should be easily extracted or recognized by nonsyntactic techniques with good accuracy. Patterns (or images of patterns) that are inherently line drawings are well represented by cursive strokes as primitives. Character recognition, chromosome classification, and waveform representa-

tion fall in this category. Examples of other primitives are polygonal regions. Representation of patterns and pattern categories rely on the theory of formal languages. A phrase-structure grammar  $G$  is a four-tuple;  $G = (V_N, V_T, P, S)$  in which  $V_N$  and  $V_T$  are nonterminal and terminal symbols of the vocabulary, respectively, of  $G$ .  $V_N \cap V_T = \emptyset$  and  $V_N \cup V_T = V$ ;  $P$  is a finite set of production rules, each rule denoted by  $\alpha \rightarrow \beta$ , where  $\alpha$  and  $\beta$  are strings over  $V$ , with  $\alpha$  involving at least one symbol from  $V_N$ .  $S \in V_N$  is the start symbol. Grammars are classified by restrictions on the types of production rules, which in turn restrict the sets of sentences that can be generated. No restrictions on  $P$  implies an unrestricted grammar and the set of sentences generated by an unrestricted grammar is called the unrestricted language. Context-sensitive grammars allow production rules of the form  $\gamma_1 \alpha \gamma_2 \rightarrow \gamma_1 \beta \gamma_2$ , where  $\alpha \in V_N$ ,  $\gamma_1, \gamma_2$ , and  $\beta$  are strings formed from the total vocabulary. The interpretation of such a production rule is that  $\alpha$  can be replaced by  $\beta$  in the context of  $\gamma_1$  and  $\gamma_2$ . Context-free grammars have production rules in which a nonterminal symbol may be replaced by a non-null string.

Consider the patterns generated from four primitives corresponding to unit-length arrows in the north, east, south, and west directions (Fig. 6). It may be of interest to recognize square patterns against other polygons generated by the same set of primitives. Sentences of the form  $\alpha^n b^n c^n d^n$  represent a set of such squares, although there are other sentences representing squares in the same model ( $\alpha^n$  is the short-form notation for the concatenation of  $n$  symbols of  $\alpha$ ). Further, such patterns may be generated by context-free as well as context-sensitive grammars. Therefore, primitive selection and grammar construction should be carried out in an interactive, iterative development. Recognition that a syntactic pattern belongs to a particular language (class) is performed by syntax analysis or parsing. The parser will also derive the process through which the syntax of the pattern is generated, thus giving us the description of the pattern. The top-down parsing (qv) method starts with the start symbol  $S$  and applies production rules of grammars of different categories to attempt generation of the given sentence (see Processing, bottom-up and top-down). This goal-oriented approach sets intermediate goals and changes them when they fail to be satisfied. In the bottom-up parsing procedure production rules are applied backward to reduce the sentence to the start symbol. At any stage the string is searched for existence of substrings that are right parts of allowed production rules. Several efficient parsing algorithms have been developed for the many kinds of

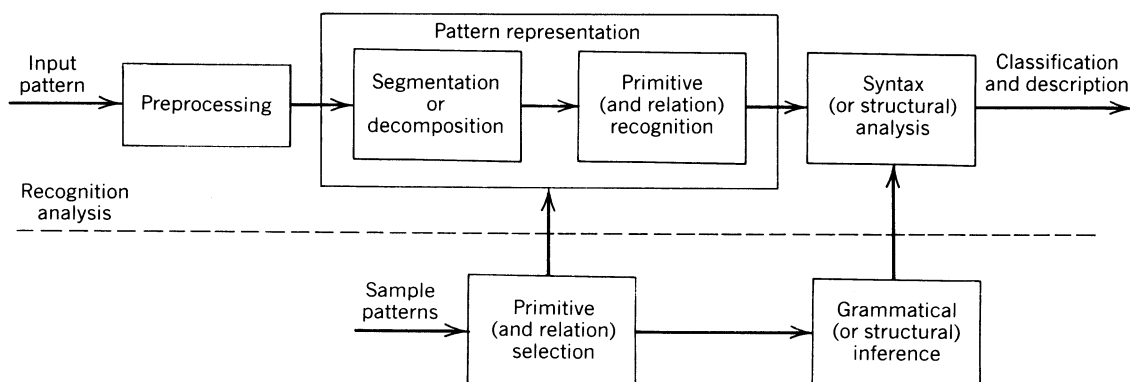


Figure 5. Development of syntactic pattern-recognition system.

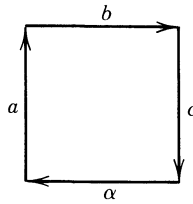


Figure 6. Set of primitives to generate class of polygons.

grammars used in syntactic pattern recognition. In practical situations in which possible sentences in a grammar category overlap with those in another category, stochastic grammars are useful to model the associated uncertainty. They are also useful when a grammar constructed to generate desired patterns is able to generate unwanted patterns in addition. A probability is associated with every string in a language. Multiple productions are handled by randomizing the productions and the state transitions in the recognition procedures. Another way of handling such ambiguities is by defining similarity measures over pairs of sentences and deciding the category with the maximum similarity.

Whereas strings of primitives are powerful in representing many syntactic patterns, there are operations on primitives more general than simple concatenation. For example, if one wants to indicate the topographic relation between three or more primitives, concatenation will not suffice. Web and graph grammars are important high-dimensional grammars. Tree grammars are a very important class of graph grammar and are naturally applicable whenever patterns can be described by trees. An application of tree grammars is in representing patterns of electrical networks constructed from elemental building blocks of two-port networks. The syntax analysis or the recognition procedure for patterns generated by tree grammars is carried out by tree automata (11).

Grammatical inference is the process by which training data are analyzed to work out the grammars that generate the required patterns. Care is required to limit the number of unwanted patterns generated by an inferred grammar as well as to avoid generating the same sentence by grammars of two or more categories (unless the training data itself had identical patterns belonging to different categories). The mathematical problem is to infer syntactic rules of an unknown grammar  $G$  based on a finite set of sentences from a language. Another set of sentences generated from grammars of other categories may also be given. In most cases there is no unique grammar as the solution and hence no unique technique for inference. Inference (qv) techniques are simplified by making extra assumptions. Some of these techniques are grammatical inference via a lattice structure, by using structural information sequence, and through inductive inference. These are discussed in Fu (11).

**Problem-Reduction Representations.** The state-space search methods outlined earlier are an application of an AI approach to pattern recognition. Another AI model found useful for pattern recognition is the problem-reduction representation (PRR). This approach establishes subproblems and sub-subproblems until, finally, the original problem is reduced to a set of primitive solvable problems. The overall problem to be solved (the recognition of a pattern to a constituent category in this case) has several (possibly only one) versions initially, only one of which is to be solved. A successor function allows

us to break up the problem into several subproblems, known as AND successors (or AND problems), all of which need to be solved before it can be claimed that the original problem is solved. Any problem can also have several versions, called the OR successors, such that the solution of any of these successors guarantees the solution of the subproblem under consideration. Possible problem reductions of an overall problem are depicted by an AND/OR graph (qv). Any AND/OR graph can be modified so that a node that is an AND problem has only OR successors and vice versa. Given the problem-reduction representations of pattern categories, the task of pattern recognition is to find the solution tree of an AND/OR graph. The tip nodes in the PRR denote primitive problems of the structural pattern recognition. It is known that context-free grammars and AND/OR trees (with a specified ordering among each set of AND successors) are equivalent. Thus, syntactic pattern-recognition problems involving context-free grammars are amenable to solution by the AI problem-reduction approach. It is useful to take this approach when solutions of primitive problems (recognition of primitives in the environment) are computationally complex. Furthermore, the AI approach brings in interaction between the detection of primitives and the description of structure. The solution tree of the AND/OR graph is searched for by a state-space search procedure called the SSS\* algorithm, which produces a rank-ordered set of solution graphs representing alternative structural descriptions of the waveform being analyzed. By using both top-down and bottom-up state-space operators, the algorithm allows a model-directed, data-confirmed and data-directed, model-confirmed approach to pattern analysis such that the occurrence of a piece of information in the data triggers the generation of model-based hypotheses. These hypotheses are then used to intelligently localize and direct the search for more information in the data space. The hypothesize-and-detect process is continued until the original problem of getting structural descriptions of the waveform is solved. In this way several lines of competing reasoning are simultaneously developed leading to multiple merit-ordered solutions. Also, data primitives can be searched for in a non-left-right manner, different from that represented by the scan order. Thus, the top-down, bottom-up, non-left-right implementation of the SSS\* algorithm permits treatment of pattern analysis in terms of searching two spaces—a model space and a data space with feedback between the two. It also allows the use of simultaneous statistical and structural information. Mathematical details are found in Refs. 22, 35, and 36. The model is well suited for recognition of waveforms among different categories based on the morphs made up of constrained mathematical curves.

**Performance Assessment and Feature Selection.** Evaluation of performance of pattern classifiers is a thorny problem. If the pattern-recognition problem is posed as an optimization problem, as it is in Bayesian decision theory, the expected performance may be computed either by analytical or by numerical techniques. However, such a measure is representative of the actual performance only if the underlying assumptions, such as forms of distributions, are true. The discrepancy between the calculated and the actual performance depends on the accuracy of the values of the parameters used. Performance of a classifier designed using a finite set of labeled samples is harder to evaluate. It is possible to design a classifier to correctly classify all the available training samples. But then

there is no way to evaluate the performance simply because there are no more labeled samples to test the classifier. Therefore, designers split an available labeled sample set into a design set and a test set. However, the total sample set itself is somewhat inadequate in practice to design a good classifier, and the designer prefers not to sacrifice a sizable portion of it. Another technique that yields a performance closer to the true value is the leave-one-out method. This requires the design of a classifier as many times as the size of the sample set, each time leaving one sample as a test sample. The average of the performance of all these classifiers, each tested with one sample, is taken as the performance. Obviously, the procedure requires excessive computation. The leave-one-out method can be generalized to the rotation method in which different numbers of samples are left out from the design set to be used for testing. Several classifiers with different combinations of design and test samples are used to obtain the average rotation-method performance.

The variation of performance of a classifier with respect to the number of features used is another important concern. Again, if the model, the forms of distributions, and the values of parameters are correct, addition of a feature cannot worsen the performance of a classifier and in general improves it. However, due to the discrepancy between the model and reality and due to the effect of finite design samples, addition of a feature can worsen the performance of a practical classifier (37). Selection of features for use in a pattern classifier is therefore a nontrivial problem. Even in the absence of any finite design samples' effect, a union of individually good features is not necessarily a good choice for a proper subset of features of a specified size. Thus, to select the best set of  $n$  features from a larger set of  $N$ , the only guaranteed technique is the exhaustive search involving the enumeration of the performance of all classifiers each designed with one subset of  $n$  features. The problem of optimal classifier design is compounded by the above-mentioned effect of finite design samples. However, in practice, it is being increasingly recognized that globally optimal pattern classifiers are neither necessary nor beneficial for applications of short duration. The user is willing to sacrifice optimality in favor of easier design and operation. Nevertheless, an understanding of theoretical issues in feature selection, dimensionality, and finite sample effects helps in better design. Toussaint (23) is a good review on error estimation. More recent work on error estimation and effects of dimensionality and sample size can be found in a number of articles in Krishnaiah and Kanal (25). See especially the articles by van Campenhout and by Jain and Chandrasekharan in Ref. 25.

### Applications

Pattern recognition has been applied to real-life problems in a wide variety of areas including medical information processing, speech processing, character recognition, fingerprint identification, earth and space sciences, particle physics, and chemistry. A brief mention of some specific problems and dominant approaches to them follows.

**Medical Information Processing.** Identification of diseases and affected areas from chest X-rays and CAT (computer-aided tomography) scans have been attempted by pattern-recognition techniques. Clustering is a common approach to these problems. Automatic recognition of white blood cells from

their images is an active area. The requirement is in computing the percentages of various types of white blood cells present in a sample of peripheral blood. Several tens of features are formulated with images in two color bands. Interactive and decision-tree approaches are used in addition to conventional methods. Often, 85–90% correct recognition rates are reported (38). Syntactic pattern recognition has been successfully applied to the classification of chromosomes from their images using cursive strokes as primitives. Medical waveforms processed by pattern recognition are electrocardiograms (ECGs), electroencephalograms (EEGs), carotid pulse waves, and heart murmurs. Syntactic techniques have been used to classify ECGs into normal and different kinds of abnormality. A signal-processing technique, deconvolution, has been attempted to separate the excitation waveform and the effect of filtering it goes through before being sensed by electrodes. Together the two help in better identification of the state of the vascular system. Statistical techniques for ECG analysis are reviewed in Ref. 40. The theory of random processes applied to ECGs has helped in classifying different stages of sleep. Carotid pulse waves are sensed by noninvasive techniques and are known to help in identifying the state of cardiac activity. Studies on structural representations of carotid pulse waves using the PRR approach appear in Refs. 22, 35, and 36. A set of original carotid pulse waveforms showing several variations is drawn in Figure 7.

**Speech Processing.** Recognition and understanding of spoken information initially received wide attention but is now limited to specific problems within the general area (34). The most widely used speech recognizers are limited vocabulary recognizers of vocabulary sizes as large as a thousand words

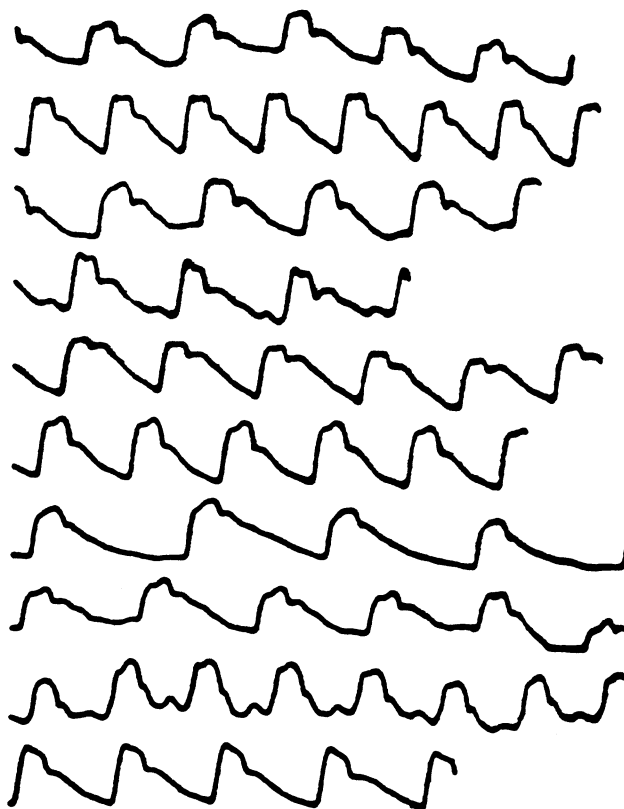


Figure 7. Examples of carotid pulse wave segments.

(see Speech understanding). The main approach continues to be segmentation (intervals of a specified size; 10–20 ms each), computation of a distance value and template matching of the sequence with those of stored templates by dynamic warping. Continuous speech-recognition techniques are reviewed in Ref. 41. Speaker recognition received some attention with possible applications in security checks. But practically achievable accuracies fall below acceptable levels. However, the techniques are useful in speaker normalization, a training procedure to tune speech-recognition systems to one of several speakers.

**Earth and Space Sciences.** Classification of remotely sensed data (from airborne and spaceborne platforms) is useful in forest and crop inventory, monitoring seasonal changes, water pollution, and in detection of earth resources. Typically, a scanner senses intensities of electromagnetic waves radiated/reflected from the earth's surface in several spectral bands. Bayesian approaches require ground truth. Parameter selection, feature selection, design of discriminant functions, and performance evaluation are the steps in the iterative development of these classifiers. Clustering and decision-tree approaches have also been widely used. There have also been attempts to use syntactic pattern recognition using textural features. A review of pattern-recognition techniques for remote-sensing applications appears in Ref. 42. Other earth sciences' areas for application of pattern recognition are meteorology, oil exploration through geophysical signal processing, and oil-spill identification. In meteorology, data gathered from satellites as well as wind speeds and directions, pressure, humidity, etc., are correlated to predict weather patterns by statistical methods. In geophysics, acoustic waves (usually impulses) are generated by an artificial explosion under the earth's surface. As the waves propagate, their shapes get modified; they are also reflected at junctions of two different kinds of materials. Reflected data are gathered by several sensors. Statistical time-series and pattern-recognition techniques are used to locate the desired junctions within the earth. The problem in oil-spill identification is the matching of spilled oil samples from coastal water with one or more suspect oil samples to detect the spiller. Matching is usually accomplished by interactive pattern recognition and graphical displays of multidimensional data extracted from the oil sample by infrared and fluorescent spectroscopy (43).

**Fingerprint and Character Recognition.** Fingerprints are powerful in identifying a human being. The need for automatic fingerprint recognition in security applications arises due to the large number of templates with which a sample should be compared. Recently, syntactic techniques with tree grammars have been applied to this problem (11). Primitives used are directional strokes (in several directions) merging and branch points, loops, segment, and abrupt ending. In character recognition the typical requirement is the classification into the alphabet of a binary matrix obtained by digitizing a character. Recognition of machine-printed or typewritten text is much simpler once the font is known. Handprinted character recognition (qv) has applications in the post office. Both statistical and syntactic techniques have been used for this purpose. The simplest technique assumes as many binary features as the number of pixels in the matrix and class conditional independence among features. Sequential and decision-tree approaches have also been formulated (44). The syntactic

method is especially suited for Chinese character recognition as they are made up of stroke segments. An overall system described in Fu (11) consists of a contour tracing program to determine the boundary of each character component. A search is then conducted to find a stroke segment to be used as a starting point. The algorithm then crawls along the stroke until the end of the stroke or a junction of strokes is encountered. The graph of the component is then built in terms of the stroke segments. The graph is traversed in a specific order, and a sequence of primitives is generated. Final recognition is achieved by matching the pattern with syntactic representations of characters.

**Radar and Sonar.** The most common pattern-recognition problem in radar and sonar is the two-class problem of deciding the presence or absence of a target based on noisy signals received from target areas. Radar radiates electromagnetic waves and processes the reflected signal. Radars are used to detect aircraft and ship targets from sensors stationed on ground, ships, and aircraft. Sonars use acoustic waves and are only useful underwater. Passive sonars rely on acoustic noise generated by ships' machinery for target detection. The set of features used for classification by a radar is known as a radar signature, a vector sequence of temporally ordered returns from the target. Parametric classification using estimated parameters and nonparametric and sequential schemes are widely used. In the sequential technique decisions are defined on successive signatures. An example of a multiclass radar problem is the classification of ship types. Radar applications of pattern recognition are reviewed in Ref. 45.

Other examples of applications of pattern recognition are analysis of photographic plates exposed in particle cloud chambers, detection of materials through nuclear magnetic resonance, identification of compounds by chemical means, and classification of rocks.

## CONCLUSION

Automatic pattern recognition has been attracting the attention of investigators from engineering, science, the humanities, and the arts. The methodologies and applications briefly discussed in this entry, written for the nonspecialist, are undoubtedly only the beginning of many exciting developments likely to occur because of the interdisciplinary nature of the subject.

## BIBLIOGRAPHY

1. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
2. K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, New York, 1972.
3. J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley, Reading, MA, 1974.
4. J. Sklansky and G. N. Wassel, *Pattern Classifiers and Trainable Machines*, Springer-Verlag, New York, 1981.
5. P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
6. S.-T. Bow, *Pattern Recognition*, Marcel Dekker, New York, 1984.
7. M. R. Anderberg, *Cluster Analysis for Applications*, Academic Press, New York, 1973.
8. J. A. Hartigan, *Clustering Algorithms*, Wiley, New York, 1975.



9. J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum, New York, 1981.
10. A. Kandel, *Fuzzy Techniques in Pattern Recognition*, Wiley, New York, 1982.
11. K.-S. Fu, *Syntactic Pattern Recognition and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
12. T. Pavlidis, *Structural Pattern Recognition*, Springer-Verlag, Berlin, 1977.
13. R. C. Gonzalez and M. G. Thomason, *Syntactic Pattern Recognition*, Addison-Wesley, Reading, MA, 1978.
14. F. Mosteller and J. W. Tukey, *Data Analysis and Regression*, Addison-Wesley, Reading, MA, 1977.
15. D. D. Wolff and M. L. Parsons, *Pattern Recognition Approaches to Data Interpretation*, Plenum, New York, 1983.
16. D. R. McNeil, *Interactive Data Analysis*, Wiley, New York, 1977.
17. Y.-T. Chien, *Interactive Pattern Recognition*, Marcel Dekker, New York, 1978.
18. G. Nagy, "State of the art in pattern recognition," *Proc. IEEE* **56**, 836–862 (May 1968).
19. Y. C. Ho and A. K. Agrawala, "On pattern classification algorithms: Introduction and survey," *Proc. IEEE* **56**, 2101–2114 (December 1968).
20. L. Kanal, "Interactive pattern analysis and classification systems: A survey and commentary," *Proc. IEEE* **60**, 1200–1215 (October 1972).
21. L. Kanal, "Patterns in pattern recognition," *IEEE Trans. Inf. Theor.* **IT-20**, 697–722 (November 1974).
22. L. N. Kanal, "Problem-solving methods and search strategies for pattern recognition," *IEEE Trans. Patt. Anal. Machine Intell.* **PAMI-1**, 193–201 (April 1979).
23. G. T. Toussaint, "Bibliography on estimation of misclassification," *IEEE Trans. Inf. Theor.* **IT-20**, 472–479 (July 1974).
24. K. S. Fu and A. Rosenfeld, "Pattern recognition and computer vision," *Computer* **17**, 274–282 (October 1984).
25. P. R. Krishnaiah and L. N. Kanal, *Handbook of Statistics, Vol. 2, Classification, Pattern Recognition, and Reduction of Dimensionality*, North-Holland, Amsterdam, 1982.
26. T. Y. Young and K.-S. Fu, *Handbook of Pattern Recognition and Image Processing*, Academic Press, New York, 1985.
27. L. N. Kanal and A. Rosenfeld, *Machine Intelligence and Pattern Recognition* (formerly titled *Progress in Pattern Recognition*), Vols. 1 and 2, North-Holland, Amsterdam, 1981 and 1985.
28. E. S. Gelsema and L. N. Kanal, *Pattern Recognition in Practice*, Vols. 1 and 2, North-Holland, Amsterdam, 1980 and 1985.
29. G. T. Toussaint, *Computational Geometry*, North-Holland, Amsterdam, 1985.
30. A. Rosenfeld, *Techniques for 3-D Machine Perception*, North-Holland, Amsterdam, 1985.
31. M. D. Srinath and P. K. Rajasekharan, *An Introduction to Statistical Signal Processing with Applications*, Wiley, New York, 1979.
32. G. R. Dattatreya and V. V. S. Sarma, "Bayesian and decision tree approaches for pattern recognition including feature measurement costs," *IEEE Trans. Patt. Anal. Machine Intell.* **PAMI-3**, 293–298 (May 1981).
33. G. R. Dattatreya and L. N. Kanal, "Decision trees in pattern recognition," in L. N. Kanal and A. Rosenfeld (eds.), *Progress in Pattern Recognition 2*, North-Holland, Amsterdam, 1985.
34. A. V. Kulkarni and L. N. Kanal, "Admissible Search Strategies for Parametric and Nonparametric Hierarchical Classifiers," in *Proceedings of the Fourth International Joint Conference on Pattern Recognition*, Kyoto, Japan, 1978, IEEE, New York, pp. 238–248.
35. G. C. Stockman and L. N. Kanal, "Problem reduction representation for the linguistic analysis of waveforms," *IEEE Trans. Patt. Anal. Machine Intell.* **PAMI-5**, 287–298 (May 1983).
36. L. N. Kanal and G. R. Dattatreya, "Problem Solving Methods for Pattern Recognition," in T. Y. Young and K.-S. Fu (eds.), *Handbook of Pattern Recognition and Image Processing*, Academic Press, New York, 1985.
37. D. H. Foley, "Considerations of sample and feature size," *IEEE Trans. Inf. Theor.* **IT-18**, 618–626 (September 1972).
38. E. S. Gelsema and G. H. Landweerd, "White blood cell recognition," in Ref. 25, pp. 595–608.
39. G. M. White, "Speech recognition: A tutorial review," *Computer* **9**, 40–53 (May 1976).
40. J. H. van Bommel, "Recognition of electrocardiographic patterns," in Ref. 25, pp. 501–526.
41. F. Jelenik, R. L. Mercer, and L. R. Bahl, "Continuous speech recognition: Statistical methods," in Ref. 25, pp. 549–574.
42. P. H. Swain, "Pattern recognition techniques for remote sensing applications," in Ref. 25, pp. 609–620.
43. Y. T. Chien and T. J. Killeen, "Computer and statistical considerations for oil spill identification," in Ref. 25, pp. 651–672.
44. G. Nagy, "Optical character recognition: Theory and practice," in Ref. 25, pp. 621–650.
45. A. A. Grometstein and W. H. Schoendorf, "Applications of pattern recognition in radar," in Ref. 25, pp. 575–594.

L. N. KANAL AND G. R. DATTATREYA  
University of Maryland

## PERCEPTRONS

Perceptrons are a fairly typical chapter in the early history of AI but deserve special attention because the issues were eventually tied up and elucidated in a particularly tidy way. In the late 1950s it was quite common to think of self-organizing, randomly connected networks as a basis for intelligence. (The perceptron is a device that weighs evidence obtained from many small experiments in order to decide whether an event fits a certain pattern.) Learning (qv) could take place by some kind of feedback that would eventually produce systematic and adaptive behavior in the network.

This idea was often pursued in a fuzzy, somewhat romantic spirit. One of those who turned it into a different form was Frank Rosenblatt (1). He began to construct machines using a very specific concept of randomly connected networks as a mechanism for a particular kind of intelligence. His machines used very small networks and learned to discriminate among extremely simple classes of visual stimuli by a process of repeated exposure. Nevertheless, the fact that such simple machines could do anything at all gave rise to a sense of excited hope that a complex machine could do very much more. There was little awareness of the possibility that the difficulties of more complex machines might grow faster than their increase in power. Thus, the early experiments excited hopes, skepticism, and debates.

## Practical Limitations

What is exceptional about the history of the perceptron is that simultaneously with the excitement and a certain amount of funding for construction, a mathematical theory was being developed that would lead to a particularly clear understanding of exactly what perceptrons could and could not do (2). This theory has a number of different facets that lend it interest. Most directly relevant to the hopes aroused by early percep-



trons was a very clear-cut demonstration that the particular cases in which the perceptron worked were very particular cases. There was no hope that they would generalize to a method that could, in principle, carry out all interesting visual discriminations. So, in one sense, the mathematical theory tied up the concept of the perceptron and put it to rest.

### Theoretical Issues

However, the mathematical theory led to much more than proving something to be impossible. It also led to insights about why the perceptron succeeded where it did and why it could not succeed in general. These particular insights happen to connect with some very general principles of applied mathematics. Stated in the simplest form, the perceptron can be regarded as the linear case of a much larger class of algorithms. Extensive history in physics and engineering demonstrates that the linear case works very simply and by methods that do not generalize to the greater number of nonlinear cases.

The analogy with applied mathematics and the physical sciences is very straightforward; when a problem is linear, one can mobilize very powerful methods for solving it. In the case of the perceptron, when a problem can be solved in a linear way, one can generate some extremely powerful algorithms whose results seem totally counterintuitive. An example is the design of a perceptron to count the number of "blobs" in a picture.

No perceptron can tell whether there is more than one object in a given topological scene. On the other hand, if one knows that the only kind of object that exists has a reasonably simple shape—like a circle or ellipse or octagon (any shape that does not have too many concavities)—it is possible for a perceptron to count them. The fact that a perceptron can do this is not at all obvious. Indeed, it is quite counterintuitive for someone who has general knowledge of what perceptrons can and cannot do.

The method of designing a perceptron to do this is rooted in some quite deep and not at all obvious mathematics—as are many of the uses of linear methods in physics. In fact, the possibility of these blob-counting perceptrons is another way of expressing one of the fundamental theorems of topology: Euler's theorem on the invariance of a number. This has become known as the Euler number: "Diameter-limited perceptrons cannot recognize any nontrivial topological properties except the Eulerian predicates" (3).

### As Context for Other Issues

The perceptron can also be used as a context for discussing general methodological issues. A few years ago, Piaget and Chomsky (4) were brought together at a conference in Royeaumont, France. Not surprisingly, the question came up concerning what kinds of knowledge could be innate. The position elaborated by Chomsky and Fodor (5,6) essentially insists that the basic structure of mental abilities must be innate, that there cannot be any significant learning of basic structures. Piaget's view is that most of what we observe in intelligence develops through an interaction between the organism and its environment; although certain innate qualities obviously exist, they do not have any kind of one-to-one correspondence to the abilities of grown-up, developed, intelligent individuals.

Perceptrons entered this debate in the following way. Sup-

pose one has an automaton organized as a perceptron and asks if the ability to do arithmetic, the idea of number, is innate to this machine. Clearly, there is an obvious sense in which the idea of number is not innate. One could very carefully examine the specifications of the machine for a long time without finding any representation of number there. Yet the Euler theorem tells us that something structured in this particular, linear way is matched in a rather specific way with the ability to count. That is, the machine is able to learn to count because it has this structure, even though the structure itself is very far removed from the structure of numbers. This understanding of the perceptron challenged both parties to clarify what they meant by *innate*.

Thus, the perceptron can be seen as valuable in three ways. First, the theoretical understanding of the perceptron had practical consequences. Pattern-recognizing devices (see Pattern recognition) of any significant size simply cannot be built that way, so new generations of AI researchers are pursuing other possibilities. Second, certain theoretical issues emerged from these efforts to carry out functions such as vision (qv), thinking, or learning (qv). The perceptron turned out to be the linear case of a much more general process. Third, the perceptron can be used to sharpen understanding of quite different issues. Because it is clear what the perceptron can and cannot do, other issues can be crystallized by posing them in this context—one in which divergent positions must be stated more rigorously.

### BIBLIOGRAPHY

1. I. F. Rosenblatt, "The perceptron: A perceiving and recognizing automaton," Rep. 85-460-1, Project Para, Cornell Aeronautical Laboratory, Inthaca, NY, 1957; *ibid.*, *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*, Spartan Books, Washington, D.C., 1962.
2. M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, 1969.
3. Reference 2, p. 134.
4. *Language and Learning: Debate Between Piaget and Chomsky*, M. Piattelli-Palmarini (ed.), Harvard University Press, Cambridge, MA, 1980.
5. N. Chomsky, *Reflections on Language*, Pantheon Books, New York, 1975.
6. J. A. Fodor, *The Language of Thought*, Harvard University Press, Cambridge, MA, 1979.

S. PAPERT  
MIT

**PERFORMANCE LINGUISTICS.** See Linguistics, competence and performance.

### PHENOMENOLOGY

As an approach to philosophical issues and problems, phenomenology splits into two very different and opposed points of view. It will be useful to identify these, at least briefly, because both have been brought to bear, though in very different ways, on the philosophical analysis of recent work in AI.

Both branches of phenomenology share a common philo-

sophical ancestor, Edmund Husserl (1). Husserl's aim was to turn philosophy into a strict science by identifying its proper subject matter and method. Husserl believed that philosophy was essentially concerned with the understanding of intentionality. This notion is notoriously slippery, but for present purposes only the roughest idea of what is involved is needed. Intentionality includes such things as having meaning, being about or representing something. For Husserl, following Brentano (2), intentionality was not primarily a property of signs or symbols but of conscious thought. The intentionality of signs and symbols was to be construed as secondary, derived from the intentionality of the conscious acts in which they figured. And intentionality was not just one property of consciousness among others, according to Husserl, but was its defining characteristic. On this view conscious acts are by their very nature of or directed toward some represented objective.

Husserl's theory of intentionality may be thought of as a response to certain problems faced by Brentano. Brentano accounted for the directedness of conscious acts by means of their objects. For Brentano, to say that conscious acts are intentional is to say that they always have an object. But it is difficult to account in this way for acts that do not seem to reach their objectives, e.g., illusory perceptions or hallucinations of non-existent entities. For Husserl, the directedness of consciousness is not a matter of the objects of its acts, which may or may not exist in any particular case, but rather of the internal content that makes them the kind of act they are and determines the sort of object required to fulfill them.

Brentano's thesis that every mental act has an object thus became, for Husserl, the thesis that every act has a meaning, or *noema*. The directedness of mental acts is a function of their internal sense whether or not they have any external reference. And Husserl devoted the remainder of his philosophical career to the investigation of experience in terms of this model of conscious acts.

### Transcendental and Existential Phenomenology

The split in phenomenology began during Husserl's lifetime with the publication of Heidegger's *Being and Time* in 1927 (3). Heidegger disagreed with Husserl over the fundamental nature of intentionality and, consequently, over the appropriate analysis of human experience. The philosophical legacy of this division is orthodox Husserlian (sometimes called "transcendental") phenomenology on the one hand and existential phenomenology on the other; both versions are alive and well today.

Husserl believed that intentionality was to be understood in terms of the necessary structures by means of which consciousness organizes experience. He was convinced that these structures were determined by the nature of consciousness itself, so that from the appropriate standpoint one could see not just how things happened to be but how they must be in terms of the conditions governing the possibility of conscious experience of the relevant sort. So Husserl sought the laws determining the experience of objects as objects in general, as perceptual objects, as objects with cultural significance or instrumental value, etc. Since these laws are implicit in individual conscious experience, the individual could gain reflective access to them if certain elaborate methodological precautions were taken.

The distinguishing features of this theory of intentionality vis-à-vis Heidegger's are the following. Individual conscious-

ness is the primary subject of intentionality. Meaning or intentional content consists of a number of layers arranged in a natural hierarchy. The more fundamental levels of meaning represent objects simply as objects of perception or conscious apprehension. Higher layers of meaning add more complicated characteristics such as those objects have as practically useful or valuable things.

For Heidegger the primary subject of intentionality is social rather than individual. Human existence is culturally conditioned, and meaning is a function of a public network of norms and practices rather than a matter of private mental contents. As a result, the most basic level of meaning includes essentially the practical utility and value of things, and things as mere objects of perception or conscious apprehension emerge only as the result of a special effort to strip away their more fundamental characteristics for legitimate practical purposes (as in science) or perverted philosophical ones (as in traditional metaphysics). In short, Husserl's hierarchy of meaning is turned upside down and his theory of intentionality inverted.

### Phenomenology and AI

What bearing has any of this on work in AI? More than two decades ago philosophical critics of cognitivism in general and AI in particular began to appeal to the work of Heidegger, Merleau-Ponty (4), and other existential phenomenologists in support of their positions (5). This led to a closer look at Husserl in terms of the current philosophical debate over the foundations of cognitive science (qv). The result was a number of surprising and still controversial claims (6), including the following:

Husserl should be viewed as an interesting precursor of modern cognitive science;

the problems with which Husserl wrestled are essentially the problems plaguing current researchers in AI; and

the debate between Husserl and Heidegger over the nature of intentionality is virtually identical to the current debate over the possibility of AI.

Rather than examining arguments for or against these claims, I will follow the consequences they have been taken to have, from the standpoint of orthodox Husserlian phenomenology, in terms of the prospects for AI. These are as follows:

Husserl's defense of his method and fundamental assumptions demonstrates that the entire enterprise of modern cognitive science rests on a firm foundation.

Husserl's successful solutions of the problems encountered in applying his method to various forms of human experience show that the present problems facing AI have solutions within the framework in which they are currently posed.

Husserl's successful integration of the everyday world (the "lifeworld") into the framework of transcendental phenomenology in his last works shows that the phenomena associated with practical activity and everyday life require no radical Heideggerian or anticognitivist interpretations of the sort that would be fatal to the AI program.

Of course, from the standpoint of contemporary existential phenomenology, Husserl's failure in each of these areas car-

ries the appropriate negative forecast for the future of AI along with an explanation of several past and present impasses.

The plausibility of the above positions, both pro and con, emerges from a closer look at Husserl and his philosophical quarrel with Heidegger. Husserl's phenomenology hooks up with modern cognitive science in the following way. Husserl's meanings, or *noemata*, are remarkably similar both in content and function to the representations of contemporary functionalist accounts of the mental. Husserl's meanings consist of layers of "predicate-senses" with hierarchical relations of dependence among them and further components that connect them with other closely related meanings. They function as "strict rules" for organizing and unifying experience. And Husserl's technique for studying these meanings or representations, his famous "phenomenological reduction," places essentially the same constraints on the study of the mental as does what Putnam and Fodor have labeled "methodological solipsism" (7). (Of course, the motives are somewhat different. Husserl is afraid of the adverse effects of naturalism in philosophy, whereas Fodor is convinced that in the absence of a complete physical theory and an adequate theory of reference in the foreseeable future, this constraint is unavoidable.)

The very close connection between Husserl and AI becomes apparent upon closer inspection of the meanings or representational structures that give each conscious act its identity. For each object of consciousness there must be such a representational structure functioning as a rule for organizing the experience of it (8):

*Any object whatever . . . points to a structure . . . that is governed by a rule . . . , a universal rule governing possible other consciousnesses of it as identical. . . . And naturally the same is true of any "imaginable" object, anything conceivable as something intended.*

So all conscious activity is a function of representations, and that functioning is strictly rule-governed. Husserlian phenomenology, like AI, is committed to capturing the representational structures and mental operations involved in all forms of intelligent behavior.

Early attempts by AI researchers to model the human ability to cope with a world of objects made no use of expectations. Intelligence was thought of as a passive receiver of context-free facts. Husserl, on the other hand, viewed intelligent experience as a context-determined, goal-directed activity—a kind of search for anticipated facts. For Husserl, the representational structure for each object contains components that provide a context, or "horizon," of "predelineations" for interpreting incoming data. These expectations are of several kinds: those features that must remain "inviolably the same as long as the objectivity remains intended as this one and of this kind," further features that are possible or likely but not necessary for this type of object, and somewhat open indications of further related objects typically encountered along with this particular one (9). In 1973 Minsky, in a paper described by Winston as "the ancestor of a wave of progress in AI" (10), introduced a new data structure for representing knowledge that is unmistakably Husserlian in character (11):

*A frame is a data-structure for representing a stereotyped situation. . . . We can think of a frame as a network of nodes and relations. The top levels of a frame are fixed, and represent things that are always true about the supposed situation. The*

*lower levels have terminals . . . (which can specify conditions (their) assignments must meet. . . .*

*Much of the phenomenological power of the theory hinges on the inclusion of expectations and other kinds of presumptions. A frame's terminals are normally already filled with "default" assignments (see Frame theory).*

### Husserl, Heidegger, and AI

Husserl's attempt to give a complete account of the mental in terms of explicit rules and representations encountered serious problems. What Husserl came to see, in part through Heidegger's "help," was that the significance of any part of the mental involved other parts and eventually opened onto the entire world of everyday life. Husserl saw this as a difficulty of size and complexity, with which he wrestled for the remainder of his life. Heidegger saw it as a much more fundamental difficulty, indicating the incorrectness of Husserl's basic assumptions about the nature of human intentionality. So Husserl spent much of his last writings, especially the posthumously published *Crisis of European Sciences and Transcendental Phenomenology* (12), attempting to bring the world of daily life into his phenomenology by describing the representational structures through which everyday objects acquire their significance. Heidegger's early writings were devoted to showing how the nonrepresentable background of everyday life gives things a significance that cannot be understood in terms of mental representations.

Heidegger's alleged insight is that one's understanding of things is rooted in one's practical activity of coping with them in the everyday world, and that this everyday world is essentially a context of socially organized purposes and human roles that cannot be represented as a set of facts. This context and one's everyday ways of functioning in it are not something human beings *know* but, through socialization, form what human beings are. Using the term "clearing" for this background of practical life, Heidegger wrote "the clearing in which present beings as such . . . can be discerned by man . . . is not an object of mental representation, but is the dominance of usage" (13).

Dreyfus, probably the leading phenomenological critic of AI, has long been arguing for this same insight and its negative implications for the future of AI research. He claims that the failure of every attempt to generalize the techniques of the impressive microworld successes of the early 1970s (e.g., Winograd's SHRDLU (qv), Evans's Analogy Problem Program, and Waltz's Scene Analysis Program) is a result of running head-on into Heidegger's nonrepresentable background of norms and practices that determines the significance of things in the everyday world (14). Impressive in the artificial domains for which they were invented, each of these programs proved incapable of yielding anything similar to human understanding or problem solving in the absence of their initial artificial restrictions. The truth, according to Dreyfus, is that microworlds are not worlds at all or, from the other side, that the various domains or regions of the everyday human world are not anything like microworlds. This insight emerged most clearly in the attempt to program children's story understanding. It was soon discovered that the "world" of even a single children's story, unlike a microworld, is not a self-contained domain and cannot be treated independently of the larger everyday world onto which it opens. Everyday understanding seems to be presupposed in every real domain, no matter how small. The everyday world is not composed of smaller indepen-

dent worlds at all, is not like a building made of tiny bricks, but is rather a whole somehow present in each of its parts.

Husserl's response to Heidegger was to accept as much of the new perspective as possible without abandoning any of his traditional assumptions. Husserl was willing to grant that each meaning functions only against the practical horizon of the "life-world." But he was convinced that the background practices themselves must consist essentially of a system of "sedimented beliefs," each with its own meaning content, which could be reactivated by phenomenological investigation and analyzed in the orthodox Husserlian manner. AI has responded to Dreyfus's attack and the relevant research impasses in much the same manner. Surely the everyday background, the commonsense world, must be a belief system, a set of implicit assumptions that differ from explicit ones only by being more easily overlooked by the cognitive scientist and more difficult for the ordinary person to recall. This cognitivist conviction has become more difficult to maintain in the light of increasing estimates of the necessary size and complexity of the required system of beliefs as well as by the lack of progress in dealing with the "frame problem." Intelligent guesses at the number of beliefs involved in everyday knowledge have grown from Minsky's 1968 estimate—"I therefore feel that a machine will quite critically need to acquire the order of a hundred thousand elements of knowledge in order to behave with reasonable sensibility in ordinary situations. A million, if properly organized, should be enough for a very great intelligence" (15)—to Dennett's 1984 estimate—"We know trillions of things"—and his likening of the cognitive scientist's reaction to any exercise of commonsense knowledge to "the unsettling but familiar 'discovery' that so far as armchair thought can determine, a certain trick we have just observed is flat impossible" (16).

The frame problem is the problem of updating the massive belief system involved in everyday knowledge to take account of changes as time passes and actions are performed. Somehow effortless and automatic for human beings, it adds another dimension to the already unmanageable problem of programming a computer to display common sense and has led Fodor to make the following comment (17):

*If someone—a Dreyfus, for example—were to ask us why we should even suppose that the digital computer is a plausible mechanism for the simulation of global cognitive processes, the answering silence would be deafening.*

### AI and Human Expertise

The newest challenge to the working assumptions of AI draws more heavily on the existential phenomenology of Merleau-Ponty than on that of Heidegger. Merleau-Ponty held that all human behavior, including cognitive behavior, could best be understood in terms of the development and employment of habitual skills and that all such skills, from the motor or sensory-motor to the "purely" intellectual, had the same basic structure (4). In terms of this view, Heidegger's everyday background is to be understood as the world of the ensemble of skills of a human being, and those skills range from the most basic perceptual and motor skills to the most sophisticated social and intellectual ones. As was the case for Heidegger, these skills cannot be captured in terms of representations and rules, and the world of a skill or ensemble of skills is not equivalent to a belief system of any sort.

The principal assailant bringing such a view to bear on

work in AI is, to borrow Fodor's phrase, "a Dreyfus," not Hubert this time, but his brother Stuart. The principal target is the branch of AI known as "expert-systems" research, but the criticism strikes at the foundations of the whole AI enterprise. Research in expert systems (qv) attempts to endow computers with human expertise in very specific domains (e.g., medical diagnosis, spectrograph analysis, and various areas of management—Samuels (18) checkers-playing program is a well-known example of this type of approach) in the following way. Human experts in the domain are interviewed to ascertain the rules or principles they employ. These are then programmed into the computer. Human experts and computers then work from the same facts using the same inference rules. Since the computer cannot forget or overlook any of the facts, cannot make any faulty inferences, and can make correct inferences much more swiftly than the human expert, the expertise of the computer should be superior. And yet in study after study the computer proves inferior to the human experts who provide its working principles. Dreyfus claims that these results can be understood if one follows Merleau-Ponty's advice and pays careful attention to the actual process of skill acquisition and employment rather than forcing expertise into the currently popular information-processing mold. The following account of the stages of skill acquisition emerged from his study of that process among airplane pilots, chess players, automobile drivers, and adult learners of a second language (19). It was later found to fit almost perfectly data that had been gathered independently on the acquisition of clinical nursing skills (20). For the sake of brevity, the following summary of this account refers only to the chess players (see Computer chess methods).

**Stage 1: Novice.** During this first stage of skill acquisition through instruction, the novice is taught to recognize various objective facts and features relevant to the skill and acquires rules for determining what to do based on these facts and features. Relevant elements of the situation are defined so clearly and objectively for the novice that recognition of them requires no reference to the overall situation in which they occur. Such elements are, in this sense, context-free. The novice's rules are also context-free in the sense that they are simply to be applied to these context-free elements regardless of anything else that may be going on in the overall situation. For example, the novice chess player is given a formula for assigning point values to pieces independent of their position and the rule "always exchange your pieces for the opponent's if the total value of pieces captured exceeds that of pieces lost." The novice is generally not taught that there are situations in which this rule should be violated.

The novice typically lacks a coherent sense of his overall task and judges his performance primarily in terms of how well he has followed the rules he has learned. After he acquires more than just a few such rules, the exercise of this skill requires such concentration that his capacity to talk or listen to advice becomes very limited.

The mental processes of the novice are easily imitated by the digital computer. Since it can use more rules and consider more context-free elements in a given amount of time, the computer typically outperforms the novice.

**Stage 2: Advanced Beginner.** Performance reaches a barely acceptable level only after the novice has considerable experience in coping with real situations. In addition to the ability to handle more context-free facts and more sophisticated rules for dealing with them, this experience has the more important

effect of enlarging the learner's conception of the world of the skill. Through practical experience in concrete situations with elements neither instructor nor learner can define in terms of objectively recognizable context-free features, the advanced beginner learns to recognize when these elements are present. This recognition is based entirely on perceived similarity to previously experienced examples. These new features are situational rather than context-free. Rules for acting may now refer to situational as well as context-free elements. For example, the advanced chess beginner learns to recognize and avoid overextended positions and to respond to such situational aspects of board positions as a weakened king's side or a strong pawn structure even though he lacks precise objective definitional rules for their identification.

Because the advanced beginner has no context-free rules for identifying situational elements, he can communicate this ability to others only by the use of examples. Thus, the capacity to identify such features, as well as the ability to use rules that refer to them, is beyond the reach of the computer. The use of concrete examples and the ability to learn context-determined features from them, easy for humans but impossible for the computer, represents a severe limitation on computer intelligence.

**Stage 3: Competence.** As a result of increased experience, the number of recognizable elements present in concrete situations, both context-free and situational, eventually becomes overwhelming. To cope with this, the competent performer learns or is taught to view the process of decision making in a hierarchical manner. By choosing a plan and examining only the relatively small number of facts and features that are most important, given that choice, he can both simplify and improve his performance. A competent chess player (such a player would have a rating of approximately class A, which would rank him in the top 20% of the tournament players), for example, may decide, after studying his position and weighing alternatives, that he can attack his opponent's king. He would then ignore certain weaknesses in his own position and personal losses created by his attack, and the removal of pieces defending the enemy king would become salient.

The choice of a plan, although necessary, is no simple matter for the competent performer. It is not governed by an objective procedure like the context-free feature recognition of the novice. But performance at this level requires the choice of an organizing plan. And this choice radically alters the relation between the performer and his environment. For the novice and the advanced beginner, performance is entirely a matter of recognizing learned facts and features and then applying learned rules and procedures for dealing with them. Success and failure can be viewed as products of these learned elements and principles, of their adequacy or inadequacy. But the competent performer, after wrestling with the choice of a plan, feels personally responsible for, and thus emotionally involved in, the outcome of that choice. Although he both understands his initial situation and decides on a particular plan in a detached manner, he finds himself deeply involved in what transpires thereafter. A successful outcome will be very satisfying and leave a vivid memory of the chosen plan and the situation as organized in terms of that plan. Failure, also, will not be easily forgotten.

**Stage 4: Proficiency.** The novice and advanced beginner simply follow rules. The competent performer makes conscious

choices of goals and plans for achieving them after reflecting upon various alternatives. This actual decision making is detached and deliberative in nature, even though the competent performer may agonize over the selection because of his involvement in its outcome.

The proficient performer is usually very involved in his task and experiences it from a particular perspective as a result of recent previous events. As a result of having this perspective, certain features of the situation will stand out as salient, and others will recede into the background and be ignored. As further events modify these salient features, there will be a gradual change in plans, expectations, and even which features stand out as salient or important. No detached choice or deliberation is involved in this process. It seems to just happen, presumably because the proficient performer has been in similar situations in the past, and memory of them triggers plans similar to those that worked in the past and expectations of further events similar to those that occurred previously.

The proficient performer's understanding and organizing of his task is intuitive, triggered naturally and without explicit thought by his prior experience. But he will still find himself thinking analytically about what to do. During this reasoning elements that present themselves as salient due to the performer's intuitive understanding will be evaluated and combined by rule to yield decisions about the best way to manipulate the environment. The spell of involvement in the world of the skill is temporarily broken by this detached and rule-governed thinking. For example, the proficient chess player (such players are termed masters, and the roughly 400 American masters rank in the top 1% of all serious players) can recognize a very large repertoire of types of positions. Recognizing almost immediately and without conscious effort the sense of a position, he sets about calculating a move that best achieves his intuitively recognized plan. He may, for example, know that he should attack, but he must deliberate about how best to do so.

**Stage 5: Expertise.** The expert performer knows how to proceed without any detached deliberation about his situation or actions and without any conscious contemplation of alternatives. While deeply involved in coping with his environment, he does not see problems in a detached way, does not work at solving them, and does not worry about the future or devise plans. The expert's skill has become so much a part of him that he need be no more aware of it than he is of his own body in ordinary motor activity. In fact, tools or instruments become like extensions of the expert's body. Chess grandmasters (there are about two dozen players holding this rank in the United States, and they, along with about four dozen slightly weaker players called International Masters, qualify as what are here referred to as experts), for example, when engrossed in a game, can lose entirely the awareness that they are manipulating pieces on a board and see themselves instead as involved participants in a world of opportunities, threats, strengths, weaknesses, hopes, and fears. When playing rapidly, they sidestep dangers in the same automatic way that a child, himself an expert, avoids missiles in a familiar video game. In general, experts neither solve problems nor make decisions; they simply do what works. The performance of the expert is fluid, and his involvement in his task unbroken by detached deliberation or analysis.

This fluid performance of the expert is a natural extension

of the skill of the proficient performer. The proficient performer, as a result of concrete experience, develops an intuitive understanding of a large number of situations. The expert recognizes an even larger number along with the associated successful tactic or decision. When a situation is recognized, the associated course of action simultaneously presents itself to the mind of the expert performer. It has been estimated that a master chess player can distinguish roughly 50,000 types of positions. We doubtless store far more typical situations in our memories than words in our vocabularies. Consequently, these reference situations, unlike the situational elements learned by the advanced beginner, bear no names and defy complete verbal description.

The grandmaster chess player recognizes a vast repertoire of types of positions for which the desirable tactic or move becomes immediately obvious. Expert chess players can play at a rate of speed at which they must depend almost entirely on intuition and hardly at all on analysis and the comparison of alternatives without any serious degradation in their performance. In a recent experiment International Master Julio Kaplan was required to add numbers presented to him audibly at the rate of about one number per second while at the same time playing five-second-a-move chess against a slightly weaker, but master-level, player. Even with his analytical mind completely occupied with the addition, Kaplan more than held his own against the master in a series of games. Deprived of the time necessary to see problems or construct plans, Kaplan still produced fluid and coordinated play.

## Conclusions

What emerges from Dreyfus's account of human skill acquisition is a progression from the analytic, rule-governed behavior of a detached subject who consciously breaks down his environment into recognizable elements to the skilled behavior of an involved subject based on an accumulation of concrete experiences and the unconscious recognition of new situations as similar to remembered ones. The innate human ability to recognize whole current situations as similar to past ones facilitates one's acquisition of high levels of skill and seems to separate one dramatically from the artificially intelligent digital computer endowed only with context-free fact- and feature-recognition devices and with inference-making power.

The reason that the expert-systems programs fail to perform like human experts is implicit in the above account. When the interviewer elicits rules and principles from the human expert, he forces him, in effect, to revert to a much lower skill level at which rules were actually operative in determining his actions and decisions. This is why experts frequently have a great deal of trouble "recalling" the rules they use even when pressed by the interviewer. They seem more naturally to think of their field of expertise as a huge set of special cases (21). It is not surprising that systems based on principles of this sort do not capture the experts' expertise. In terms of skill level, the computer is stuck somewhere between the novice and advanced beginner and has no way of advancing beyond this stage. What has obscured this fact for so long is the tremendous memory of the computer in terms of numbers of facts and features that can be stored and the tremendous number of rules and principles it can utilize with superhuman speed and accuracy. Although its skill is of a kind that would place it below the level of the advanced beginner, its computing power makes its performance vastly superior to

that of a human being at the same skill level. But power of this kind alone is not sufficient to duplicate the intuitive ability of the human expert.

This way of looking at skilled behavior and its development does more than explain the failure of expert-systems research to achieve its intended goals. It also helps to explain the more general failure of AI to duplicate everyday knowledge or common sense which, as Heidegger argued, plays an essential role in all experience. As Heidegger also maintained, everyday knowledge is not a knowing *that*, not primarily a matter of explicit beliefs or relations between the mind and propositions, but is much more a matter of knowing *how*, of being able to cope with a world of implicit social norms, human purposes, and instrumental objects. And "know-how" and "skill" are virtually synonymous. In the central areas of everyday cognitive life one is, for the most part, an expert. One is an expert perceiver, speaker, hearer, and reader of one's native language and an expert problem solver for a wide range of everyday problems. This expertise does not mean that one does not make mistakes, but it does mean, if the Dreyfus account of skills is correct, that one's performance is entirely different in kind from that of the programmed digital computer. In each of these areas the computer is, at best, a very powerful and sophisticated beginner, competent in artificial microworlds where situational understanding and intuition have no part to play but incompetent in the real world of human expertise.

That is the present state of the "interface" between AI and phenomenology. The lines are clearly drawn, but the battle that is likely to ensue as a result of this latest attack has scarcely begun. The challenge to AI is to account not just for everyday knowledge but for human expertise in general, solely in terms of explicit rules and representations (see also Limits of AI; Philosophical questions; Reasoning, common-sense; Representation, knowledge).

## BIBLIOGRAPHY

1. E. Husserl, *Ideas: General Introduction to Pure Phenomenology*, Collier, New York, 1972.
2. F. Brentano, *The True and the Evident*, Routledge and Kegan Paul, London, 1966.
3. M. Heidegger, *Being and Time*, Harper & Row, New York, 1962.
4. M. Merleau-Ponty, *Phenomenology of Perception*, Routledge and Kegan Paul, London, 1962.
5. C. Taylor, *The Explanation of Behavior*, Routledge and Kegan Paul, London, 1964; H. Dreyfus, *What Computers Can't Do*, Harper & Row, New York, 1972.
6. H. Dreyfus, *Husserl, Intentionality and Cognitive Science*, MIT Press, Cambridge, MA, pp. 1-27, 1982.
7. J. Fodor, Methodological Solipsism Considered as a Research Strategy in Cognitive Psychology, in J. Haugeland (ed.), *Mind Design*, Bradford, Montgomery, VT, 1981.
8. E. Husserl, *Cartesian Meditations*, Nijhoff, The Hague, pp. 53-54, 1960.
9. Reference 8, p. 51; E. Husserl, *Experience and Judgment*, Northwestern University Press, Evanston, IL, pp. 125, 331, 1973.
10. P. Winston, *The Psychology of Computer Vision*, McGraw-Hill, New York, p. 16, 1975.
11. M. Minsky, A Framework for Representing Knowledge, Ref. 17, p. 96.
12. E. Husserl, *The Crisis of European Sciences and Transcendental Phenomenology*, Northwestern University Press, Evanston, IL, 1970.



13. M. Heidegger, *On the Way to Language*, Harper & Row, New York, p. 33, 1971.
14. See the preface to the revised edition of H. Dreyfus, *What Computers Can't Do*, Harper & Row, New York, 1979.
15. M. Minsky, *Semantic Information Processing*, MIT Press, Cambridge, MA, p. 26, 1968.
16. D. Dennett, Cognitive Wheels: The Frame Problem of AI, in C. Hookway (ed.), *Minds, Machines and Evolution*, Cambridge University Press, New York, pp. 134–136, 1984.
17. J. Fodor, *The Modularity of Mind*, MIT Press, Cambridge, MA, p. 129, 1983.
18. M. Minsky, "Artificial intelligence," *Scientif. Am.* 215(4), 247–260 (October 1966) and interview with Arthur Samuel, released by Stanford University News Office, April 28, 1983.
19. H. Dreyfus and S. Dreyfus, *Mind over Machine*, Macmillan, New York, Chapter 1, 1986.
20. P. Benner, *From Novice to Expert: Excellence and Power in Clinical Nursing Practice* (Addison-Wesley: Reading, MA, 1984).
21. E. Feigenbaum and P. McCorduck, *The Fifth Generation, Artificial Intelligence and Japan's Computer Challenge to the World*, Addison-Wesley, Reading, MA, p. 82, 1983.

H. HALL  
University of Delaware

## PHILOSOPHICAL QUESTIONS

The interests of philosophers and workers in AI intersect and overlap in many ways. Some philosophers have tried to use the resources of AI to shed new light on long-standing philosophical problems, and others have been vocal critics of the philosophical claims made by AI researchers. There has also been convergence. In carrying out specific projects, AI researchers have frequently been led into areas traditionally discussed and investigated by philosophers, providing new opportunities for collaborative exploration.

The philosophical impact of AI has been greatest on the philosophy of mind. AI has suggested new answers to long-standing questions about the nature of mind, led to the reformulation of traditional problems, and given birth to new controversies of its own. The mind–body problem, the mechanism free-will debate, and disputes about the nature of understanding, intentionality, and intelligence have all been transformed in substantial ways by the advent of AI.

There are also important connections between AI and other areas of philosophy as diverse as the philosophy of science, the philosophy of language, metaphysics, and epistemology (qv). (The relevance of logic is almost too obvious to mention.) AI issues in these other areas have not generated the sort of emotion-laden controversy associated with issues in the philosophy of mind, but in the opinion of some philosophers they may turn out to be of greater importance in the long run (1).

### Philosophy of Mind

At least since the seventeenth century and the rise of modern mechanistic theories of the physical world, philosophers have debated the place of mental phenomena within the mechanistic scheme. The development of modern computers and AI programs gave new impetus to these debates. For the first time it seemed possible to actually construct machines that were both undeniably mechanistic in their operations and possessed of

characteristics and abilities uniquely associated with minds. The possibility of computing machines able to play chess, prove theorems, and perhaps engage in conversation gave the mind–mechanism dispute a timely urgency.

**The Turing Test.** The mechanism question has been posed in a variety of related but independent forms, including: "Can machines think?" "Do computers have minds?" and "Is artificial intelligence really intelligence?" In his seminal 1950 paper "Computing Machinery and Intelligence," Turing considered the first of these questions and found it too meaningless to deserve discussion (2). He proposed therefore to replace it with another question that was more precise and decidable but that captured the essential issues raised by the more familiar popular formulation.

Turing proposed an imitation game to be played by a human interrogator and two unseen participants X and Y, one a human and the other a machine (see Turing test). The interrogator is able to address any questions he wishes to X and Y, and they are to respond by typewritten messages. Both the machine and the unseen human have the same objective in the game: Each tries to convince the interrogator that it is the human respondent. Turing's replacement question was posed in terms of this game, "Are there imaginable digital computers which would do well in the imitation game?" He answered the question in the affirmative and predicted that within 50 years digital computers would be able to play the imitation game well enough that an interrogator would have no more than a 70% success rate in his identification of the machine respondent after a 5-min conversation.

**Objections to Test's Criteria.** The truth value of Turing's prediction about machine success at the imitation game is obviously an empirical and not a philosophical matter. However, philosophers have challenged its adequacy as a substitute for the original "Can machines think?" query. Criticism has focused on the behavioral evidence that forms the basis of the test. The sample of behavior may seem to be too limited; a 5-min conversation appears an inadequate basis for a judgment of mentality. Such objections are not really very serious. Though the conversation is brief, the interrogator is not at all limited in his range of discussion topics. He may make inquiries about poetry, sports, music, or cuisine, and in all these areas the machine must produce plausibly humanlike responses. The test could also be modified to allow for longer conversations without altering its rationale.

The behavioral evidence has been challenged in a more substantial way by philosophers who have argued that behavioral criteria alone cannot suffice for the applicability of mental predicates. They argue that having a mind is not merely a matter of exhibiting certain patterns of verbal or nonverbal behavior but also requires that the right sort of internal processes produce the relevant behavior. Sometimes the objection is raised as a denial that the machine does exhibit the same behavior as a human (3). The machine and the human respondent may produce the same end result, a given sequence of words on the printer's output, such as "I do not have much taste for Mexican cooking." But it does not follow that they are exhibiting the same behavior. In the human case causing those words to be printed is the making of an assertion. It is a significant linguistic act produced as the result of a prior communicative intention. The machine respondent's printing of those same words would only count as an assertion if it also was produced by a similar communicative intention. Since the



interrogator has access only to the physically indistinguishable end products, he cannot determine which instances of seemingly communicative behavior are genuine assertions. But his inability to identify genuine assertions on the basis of his limited evidence in no way implies that the two respondents are both engaging in the same sorts of behaviors.

**A General Antibehaviorist Objection to the Test.** Another way of making the antibehaviorist point is by imagining a system that simulates human behavior well enough to pass the Turing test but does so as the result of internal processes that obviously involve no genuine thought or intelligence. One such example has been provided by Block (4). He imagines a device that consists primarily of an enormous memory that stores a very large but finite list of all English-language conversational exchanges up to a given length (with a further limit on the length of each utterance in the exchange). The list must be ordered in some way that allows rapid access. Confronted with an interrogator in a Turing test, the machine searches its memory to find a conversation whose first segment corresponds to the interrogator's initial question. It then prints the next utterance from that conversation. Following the interrogator's reply, it again searches for a conversation whose first three segments correspond to those in its present dialogue and prints the fourth utterance from that conversation. It continues this process up to the length of the test, which is no greater than the length of its stored conversations.

One may object that such a machine is wildly impractical and argue that no such machine could ever be built, but to do so would be to miss the philosophical point of the example. The machine is in practice an impossibility; indeed, the number of items it would have to store if the conversation length were increased might soon outstrip the number of elementary particles in the universe (5). As a thought experiment, the example is not intended as a practical suggestion for building conversation machines. It is intended to make a conceptual point about the notions of intelligence and having a mind by showing that they require more than the satisfaction of the sorts of behavioral criteria employed by the Turing test. Being intelligent or having a mind is also a matter of the internal processes that produce behavior. Such conceptual connections may be obscured by the fact that in normal human intercourse, judgments about another person's mental states are normally made on purely behavioral evidence. Someone is counted as understanding a story if he can paraphrase it and answer a suitable range of questions about it. But in such cases the other person's status as a rational thinking understanding agent is not in question, but only his mastery of the particular story at hand. As philosophers have noted, the criteria that suffice in these specific cases cannot simply be extended to other cases where the basic issue of having a mind at all is in question (6).

Some may object that the Block example (4) shows only that the internal processes underlying genuine intelligence must satisfy real-time constraints. However, Block's example is intended to make a stronger point and is taken by many to have done so. The existence of such a machine operating in real time is at least a logical (if not an engineering) possibility, and there is a strong intuition that such a device would have no genuine understanding of language.

**Functionalism and AI.** Philosophical dissatisfaction with the Turing test reflected a general rejection of behaviorism as a theory of mind. Few philosophers any longer believe that men-

tal predicates can be defined or explicated in purely behavioral terms. However, the functionalist theory of mind (7,8) that dominates present philosophical thinking, preserves some elements of the earlier behaviorism, especially in its claim that many commonsense or folk-psychological mental concepts are to be explicated at least partially in terms of their behavior-causing roles. The functionalist program has been strongly influenced by analogies drawn from computer science and AI, both in its general outlook and in several of its specific applications to problems about the nature of mind.

Functionalism as a distinctive position was developed in the 1960s (9–11) as an attempt to avoid the shortcomings of the two then most popular philosophical views of mind, behaviorism and physicalist identity theory, while retaining the strengths of each. Its central idea is that psychological states, such as desiring to be famous, believing that it will rain tomorrow, feeling a pain, or being angry, are type individuated on the basis of their causal functional roles in mediating an organism's or system's interaction with its environment. Being a pain or a belief that chalk is white is a matter of bearing appropriate causal relations to sensory inputs, behavioral outputs, and other internal states mediating the system's connections between perception and action. Items with radically different intrinsic natures can all count as instances of the same psychological kind as long as they play the same causal roles within their respective containing systems.

Functionalism thus rejects the Cartesian intuition that a mental state's psychological kind is fixed by its intrinsic, directly introspectible properties. For the functionalist it is the state's causal relations within the system that are relevant. Functionalism differs from behaviorism primarily in two respects (12). First, it treats mental states as genuine causes, as actual internal states that play a role in the production of behavior. Many behaviorists regarded mental predicates simply as abbreviated ways of talking about behavioral patterns or regularities. Attributing to someone a desire to be wealthy was for the behaviorist only a way to describe the agent's behavior, not to explain the production of that behavior by reference to an inner cause. Functionalism's realism about inner causes also accounts for its second difference from behaviorism. Functional states are defined by their relations not only to input and output but also to each other. Functionalism can thus deal with the holism of the mental and the fact that mental states normally produce behavior only in joint operation. A desire for a cold beer will produce little behavior in the absence of suitable beliefs. And some mental states, such as a belief in a law of logic, will have functional roles that almost exclusively concern their influence on internal processes, such as patterns of inferential reasoning.

**Functionalism and Physicalism.** Functionalism departs from its other philosophic ancestor, type-identity theory, in its emphasis on function as opposed to structure. Type-identity theorists had proposed to identify mental kinds with specific physical kinds (normally neurophysiological kinds) empirically found to be correlated with their occurrence (13). The property of being in pain might, for example, be identified with the property of having one's C-fibers firing. Functionalists, with the aid of insights drawn from computer science, have argued that even if as a matter of fact a given functional role is normally filled by a specific physical structure, other physical structures might in other contexts fill that same causal role (10). Type-identity theorists have erred in identifying mental states with the narrow range of physical states that fill the

relevant causal roles in human brains, thus unreasonably excluding organisms with different physiologies as well as robots and AI devices from the realm of the mental (7). This functionalist critique of the identity theory, which is known as the multiple-realization argument, was directly inspired by computer and AI analogies. The philosophical presentations of the argument allude to the fact that the same algorithm may be carried out on a wide range of physically dissimilar devices (10). The claim is also often explained by appeal to the software-hardware distinction, where again multiple realizations are possible and common.

**Intentional Stance.** AI has also had a strong influence in leading functionalists to distinguish among various levels of abstraction at which organized systems can be described and explained. Perhaps best known is Dennett's scheme (14), which is introduced in application to a chess-playing computer and includes three stances from which one may attempt to explain its behavior: the physical stance, the design stance, and the intentional stance. They involve descriptions and predictions based, respectively, on structure and physical causation, teleological function, and rational belief/desire explanation. They correspond in a rough way to what in AI might be called hardware, software, and knowledge-level descriptions. Dennett's account of the intentional stance should be of interest to AI researchers as well as philosophers since it divorces the notion of having intentional states such as beliefs and desires from any metaphysical commitments about the system's underlying substance (spiritual, organic, or electronic) and provides a practical method for determining when descriptions of a system's (or subcomponent's) behavior involve implicit attributions of rationality and mentality. Though Dennett's intentional-systems theory has come in for much philosophical criticism (16), it is a clear improvement over the casual and unregimented use of intentional terminology in AI by which it is partly inspired.

Dennett's work also provides the best example of another major application of AI resources to the functionalist program in dealing with the problem of hidden theoretical homunculi (17). Mentalistic psychology has often been faulted for implicitly relying on covert internal agents who account for regularities in external behavior by reproducing within a subpersonal component the cognitive abilities of the person that are supposedly being explained. The threat of vacuity or vicious infinite regress looms large when explanations of visual perception rely upon a mind's eye to perceive an internal object or explanations of rational action refer to an inner decider who ranks alternative courses of action. Dennett has drawn directly on work in AI to resolve this centuries-old problem. The first part of his solution is to apply the AI strategy of decomposing a complex task or function into a set of organized subtasks and then subjecting those second-level tasks to the same sort of decomposition, repeating the process until the whole organized hierarchy comes to rest on interacting components whose behavior is straightforwardly mechanizable. Using the intentional system's method for keeping track of implicit attributions of rationality and mentality, the descriptions at each level becomes progressively less and less mental as one descends the hierarchy. Dennett describes the procedure as decomposing homunculi at each level into a committee of individually dumber homunculi at the next level down until the process terminates at an "army of idiots" (17). Dennett's use of AI techniques thus answers philosophical criticism of homun-

culi by showing that their theoretical use need involve neither vacuity nor infinite regress.

**Computational Theory of Mind.** A third application of AI to functionalism involves the computational theory of mind (CTM). Functionalist philosophers have carried the analogy between AI programs and the organization of the mind one step further in suggesting not only that the mind decomposes into a hierarchical series of levels but also that the operations of the subcomponents at the underlying levels consist entirely in the computational manipulation of representational structures or formal symbols. The operations are computational in that they are governed by rules that can be completely and explicitly formulated in terms of the formal and syntactic properties of the representations (18). As representations, such structures also have semantic content, but the underlying processors that manipulate them do so solely on the basis of their forms and syntax. The CTM is intended to resolve (or dissolve) the philosophical problem of explaining how intentional content can have a causal impact on the physical world. According to the CTM, content can have causal consequences only insofar as it is mirrored in formal structure. Differences in representational content that are not reflected in formal differences detectable by internal processors can have no impact on behavior. The CTM thus employs an entirely syntactic taxonomy of internal representations and individuates psychological states solely on the basis of the formal objects to which they are related. Since facts about the social, cultural, historical, or physical environment play no direct role in the determination of computational content, the CTM is said to be methodologically solipsistic in its approach to psychology (18).

**The Anticomputationalist Critique.** To those philosophers who accept the CTM, it represents perhaps the most important application of AI theory to the philosophy of mind. However, it has provoked other philosophers to the strongest and most widely discussed criticisms of recent work in AI. The outstanding critic in this regard has been Searle, who in his influential article "Minds, Brains and Programs" (19) attempts to refute the claims made by Roger Schank and others that their script-based story-understanding programs literally "understand" the stories on which they comment and answer questions (see Scripts; Story analysis). Searle has a larger goal than merely challenging some perhaps exaggerated or premature claims about the level of present AI success. His aim is to refute the CTM and all work in AI that relies on it. His examples and arguments are meant to demonstrate that understanding (or having any other intentional state) can never be simply a matter of having the right sort of internal formal structure or being an instantiation of the right sort of computer program.

The central focus of Searle's argument is a thought experiment that has come to be known as the "Chinese room." Searle imagines himself locked in a room with three batches of Chinese writing and some sets of instructions for manipulating Chinese symbols. The rules are given in English, and Searle can carry them out though he does not understand Chinese since they specify the operations to be carried out purely in virtue of the shapes of the Chinese symbols and do not allude to their meanings. Searle carries out the instructions and passes back strings of Chinese symbols produced as the result of his operations. The reader is asked to imagine that the three batches of Chinese symbols correspond to what Schank and his

colleagues would call a script, a story and questions, that the instructions correspond to a program, and that the strings of symbols Searle produces represent conversationally appropriate answers to the questions about the story. Searle's contention is that in such a case he would satisfy all of the conditions on the basis of which a computer running Schank's program is said to understand Chinese, and yet it is intuitively obvious that in such a case he, Searle, would not understand a word of Chinese. Thus, he infers that Schank's computer is equally lacking in understanding of Chinese. Though Searle's argument bears some similarities to earlier criticisms of the Turing test, it has relevance to a far wider class of theories. It is not directed only against behaviorist views, which equate understanding with performance, but also against the CTM and all those versions of functionalism and AI that attempt to account for the mental production of behavior purely in terms of formal structures and computational rules.

**Replies to Anticomputationalist Argument.** Searle's argument has provoked a great deal of vigorous criticism, but he has displayed considerable dialectical skill in replying to his critics. It has been argued that though Searle alone would not understand Chinese, the ensemble of Searle plus instructions and batches of symbols does understand Chinese. In reply, Searle has offered a modified example in which he memorizes the batches of symbols and rules, though continuing to treat the symbols as mere uninterpreted shapes. In such a case every part of the ensemble would be internal to Searle, and yet intuitively he would not seem to understand Chinese. Some critics have conceded genuine understanding requires more than the formal ability to manipulate symbols since understanding a symbol's meaning requires the ability to relate the symbol to the nonsymbolic external world. Thus, genuine understanding would require a robot capable of perception and action as well as of merely "conversational" performance. Such AI proposals are in keeping with the functionalist view that mental kinds are determined by the causal role that a state plays in regulating purposeful interaction with the environment. Searle has also varied his example to answer these robot proposals, though perhaps with less convincing success. He imagines himself in the robot's control room, where in addition to his earlier symbolic inputs, various formal symbols appear on a screen. Those unknown to Searle are perceptual inputs; to him they are just further uninterpreted shapes to be manipulated according to rules governing only formal operations. Unknown to him, his activity produces appropriate external responses by the robot. Searle argues that he still would not understand Chinese though the robot's behavior might seem to show an understanding of how Chinese symbols relate to real-world items. Many functionalists do not share Searle's intuitions about the robot case, especially if the entire organized robot, rather than merely its Searle "component," is considered as the potential understander of Chinese. Searle's intuitions seem to rest on the absence of any subjective or experiential elements in the robot's internal processes, but it is a theoretically open question whether such processes are essential for genuine understanding. Thus, the controversy surrounding Searle's argument turns in the end on a basic conflict of fundamental intuitions.

Since the point of Searle's argument against AI has been often misunderstood, it is important to note that he is not a Cartesian dualist, and he does not attack AI in order to defend a spiritual view of mind. He is a straightforward materialist

about the mind's dependence on the causal structure of the brain. He accuses his AI opponents of an implicit or at least methodological dualism in their belief that the fundamental structure of the mind can be investigated and explained in the absence of any detailed information about the actual physical operation of the human brain.

**Antimechanism and the Gödel Argument.** The possibility of AI and the computational modeling of mind has also been attacked with philosophical arguments of quite a different sort, based not on intuitive thought experiments but on rigorous results in mathematical logic (qv). It has been argued that Kurt Gödel's theorems concerning the incompleteness of arithmetic and the limits of formal systems show that no machine or computational device can be a completely adequate model of the human mind and that human minds are fundamentally different from machines (see Completeness). Though these arguments have generated extensive philosophical discussion and debate, many of the central issues remain obscure, and it is often difficult to understand just what claims are being made or denied. The logical results are quite clear, precise, and beyond question. However, their application to questions about mechanism and the human mind remain far from obvious.

In his seminal article "Minds, Machines and Gödel" (20) Lucas appealed to the Gödel results in an attempt to show that no machine can duplicate the abilities of the human mind. Given any machine that might be thought to do so, Lucas argued that there will always be some sentence the machine cannot show to be true, which he, Lucas, can recognize and show to be true. The Gödel results figure in the following way. Gödel's first theorem states that any consistent formal system  $S$  containing an adequate axiomatization of arithmetic will be incomplete, i.e., there will be a sentence  $G$  of  $S$  such that neither it nor its negation is a theorem of  $S$ . Gödel proved this by showing that if  $S$  is an adequate axiomatization of arithmetic,  $S$  is adequate to express arithmetic statements that encode statements about its own syntax and proof relation. Thus, it is possible to construct within  $S$  a sentence  $G$  that encodes the statement that it,  $G$ , is not a theorem of  $S$ , i.e., that it is not provable in  $S$ . He also showed that if a sentence  $K$  is a theorem of  $S$ , there is another sentence of  $S$  that says or encodes that  $K$  is provable in  $S$ , and that latter sentence will also be provable in  $S$ . Thus, if  $G$  were provable in  $S$ , the sentence that says that  $G$  is provable in  $S$  would also be a theorem of  $S$ . But that sentence would be logically equivalent to the negation of  $G$ . So both  $G$  and its negation would be theorems of  $S$ , and  $S$  would not be consistent. So if  $S$  is consistent,  $G$  cannot be proved in  $S$ . It is this Gödel sentence  $G$  that plays the crucial role in Lucas's argument. He claims that any machine  $M_i$  is the concrete instantiation of a formal system  $S_i$  such that the sequence of states through which  $M_i$  passes in producing a sentence  $F$  as output corresponds to a proof of  $F$  in  $S_i$ . Thus, he argues that for any machine  $M_i$  put forward as a candidate model of the human mind there will be a Gödel sentence  $G_i$  that  $M_i$  cannot produce as true. This is the sentence that says of itself that it is not provable by  $M_i$  (or not provable in the formal system  $S_i$  which  $M_i$  instantiates). But Lucas asserts that he, standing outside of  $M_i$ , can see that  $G_i$  is true. Thus, there is something he can do that  $M_i$  is not able to do, and  $M_i$  has failed to duplicate his abilities. Since the argument presented is fully general and applies to any machine, Lucas con-

cludes that no machine can duplicate the abilities of the human mind.

**Replies to Gödel Argument.** The replies to Lucas's argument have been numerous and diverse. His claim that any machine is the instantiation of a formal system has been questioned since it is not at all clear just what counts as a machine in Lucas's sense. A more precise Lucas-style argument can be given if the well-defined notion of a Turing machine (qv) is substituted for Lucas's inexact intuitive notion of a machine. The revised claim is that no Turing machine can be an adequate model of the human mind. It is easy to link formal systems with Turing machines since for any given formal system  $S_i$  there is a Turing machine  $T_i$  whose output consists exactly of the theorems of  $S_i$ . For any such Turing machine  $T_i$ , there will be a Gödel sentence  $G_i$  it cannot prove, a sentence that says of itself that it is not provable by  $T_i$ . Though shifting to a claim about Turing machines provides a clear connection with Gödel's results about axiomatized formal systems, it leaves unclear the implications concerning actual concrete machines.

A Turing machine is an abstract device exhaustively specified by its machine table, which is a function from ordered pairs of machine state and input to ordered pairs of subsequent machine state and output. Any actual concrete physical device will be an instantiation of many different Turing machines. As Dennett (21) has argued, the limits that apply to a concrete device under one of its Turing-machine descriptions need not restrict absolutely what it can do as a physical machine or under one of its other descriptions. If an actual device  $X$  is an instantiation of Turing machine  $T_i$ , it cannot under that description prove  $G_i$ , the Gödel sentence of  $T_i$ , but  $X$  may well do so under one of its other descriptions. Thus, the Turing-machine version of Lucas's claim might well prove inadequate to establish his larger antimechanist conclusion. Nonetheless, if the Turing-machine claim could be established, it might be sufficient to refute the computational theory of mind and undermine AI hopes to duplicate or fully explain human mental abilities since any modern digital computer is in principle equivalent to some Turing machine (leaving aside the fact that the computer as physical device might never produce its total output).

However, the Turing-machine version of Lucas's argument is also open to objection. Benacerraf (22) has pressed the question of just what it is that Lucas supposes himself able to do in besting an alleged Turing machine duplicate  $T_i$ . As Benacerraf notes, it cannot be proving  $T_i$ 's Gödel sentence  $G_i$  as a theorem of  $S_i$ , that is, proving  $G_i$  using  $S_i$ 's axioms and inference rules. Lucas is no more able to do this than is  $T_i$ . But if the sense of "prove" is left vague and informal, it no longer remains certain that  $T_i$  cannot in this informal sense "prove"  $G_i$ . Another major line of criticism has focused on the fact that the Gödel result applies only to consistent formal systems. Thus, in order to establish the truth of  $G_i$  for any Turing machine  $T_i$ , Lucas must be able to show that  $T_i$  or its corresponding formal system  $S_i$  is consistent. How can Lucas know this? Given the dialectical manner in which he presents his imagined contest with the mechanist, Lucas does have a response that he can and does make on this point (20). Lucas assumes that humans are consistent. Thus, if a candidate machine is inconsistent, it cannot be an adequate model of the human mind. If it is consistent, the Gödel result applies to it, and it can be shown inadequate by the original argument. The assumption that humans are consistent, on which Lucas's re-

sponse depends, would seem to be falsified by ordinary facts and observations. Human beings are far from perfectly consistent. Lucas has tried to deny the relevance of familiar human inconsistency by distinguishing between different ways of being inconsistent: as the result of malfunctioning misuse of consistent principles and as the result of the proper use of inconsistent principles. However, Lucas's claim that humans are inconsistent only in the former sense and thus consistent in the sense needed for his argument has been found less than convincing and been regarded as one of his argument's weakest links (22). The consistency assumption is especially problematic since it is not independent of the central point at issue. If humans are Turing machines, it follows by Gödel's second theorem that they cannot prove their own consistency (at least in the sense of proving it within the formal systems corresponding to the machines they instantiate). Thus, for Lucas to assume he can prove his own consistency might seem to beg the question against his mechanist opponent.

Although Lucas's arguments fail to establish his anti-mechanist conclusions, the relevance of the Gödel results to AI remains an unresolved but intriguing issue of potentially great philosophical importance. Benacerraf has suggested that perhaps the Gödel theorems show that if humans are Turing machines, they cannot know which machines they are (22), and Hofstadter has conjectured that they may provide a fundamental key to understanding consciousness and the nature of mind (23).

### Epistemology and AI

The nature of knowledge has been a central question of philosophical investigation since the birth of philosophy. The problem originally posed by Plato of how to distinguish knowledge (episteme) from true opinion (doxa) remains a subject of debate. Though it is generally accepted today that knowledge cannot be analyzed merely as justified true belief (24), most philosophers do accept the necessity for including some sort of justification condition in the analysis of knowledge. Knowledge differs from mere true belief at least in part in the rational justification the knower has for his belief. Thus, the theory of knowledge has a major interest in the nature of human reasoning. Its concerns here naturally overlap with those of AI researchers attempting to formalize systems of rational inference for use in cognitive problem-solving programs and knowledge engineering. Philosophical assessments of the prospects for machine rationality have differed widely and tended to reflect the affinities or divergences between competing philosophical views of rationality and the methods employed by AI. On the whole, philosophers of a phenomenological orientation have been less sympathetic and more pessimistic about AI attempts at programming rationality, and philosophers in the logical empiricist tradition have been more optimistic.

### Phenomenological Critique

**Critique of early AI.** The phenomenological critique of AI's attempt to formalize human reasoning can be best understood through the work of its most prominent proponent, Hubert Dreyfus (see Phenomenology). In the two editions of his book "What Computers Can't Do" (25), Dreyfus aimed to expose the weaknesses and shortcomings of a variety of then existing AI programs. His larger intent was to show that these defects were not merely incidental to the programs he considered nor

remediable by further use of the basic techniques originally employed. Dreyfus argued that the programs were flawed in principle, failed to take account of essential features of creative human problem solving, and relied on fundamentally mistaken underlying assumptions. Dreyfus examined early AI work in game playing, language comprehension, problem solving, and pattern recognition and found that the programs in each area lacked important abilities possessed by humans. The chess-playing programs he considered rely on heuristically guided searches and examine a large number of possible moves. Human expert players are able to zero in on a small number of promising moves as the result of fringe consciousness, the phenomenon by which implicit background understanding focuses attention and transforms the object of attention. In the chess-playing case (see Computer-chess methods) the fringe consciousness embodies the expert player's implicit understanding of global patterns of board organization acquired through experience. In problem-solving (qv) programs Dreyfus focused on human insight and the ability to discriminate the essential from the nonessential features of a task situation. Understanding the deep structure of the problem is a necessary first step in human problem solving. Dreyfus noted that in many AI programs this process was carried out not by the program but by the programmer in setting up the problem task and in the choice of factors from which the program was required to fashion a solution. A similar point applies to computer learning (qv) in which the Piercean process of abduction is largely performed by the programmer. The human ability to learn is as much a matter of figuring out what factors are likely to yield regularities and how they must be categorized in order to reveal them as it is of inductively discovering the connections among those variables. However, in the AI learning programs considered by Dreyfus, such as Winston's arch-learning program (26), the range of potentially relevant factors was already greatly constrained and categorized by the programmer. Thus, such learning programs at best simulate one component of the human ability to learn, and perhaps the less interesting component. In the area of language comprehension Dreyfus emphasized the human ability to tolerate a high degree of ambiguity in linguistic expressions and to disambiguate them in context on the basis of extralinguistic information relevant to the communication situation. Moreover, he argued that this human ability did not depend on the use of any underlying fully determinate rules. In contrast, the early language-comprehension programs he considered relied on determinate rules sensitive to only a much narrower range of variables actually present in the text.

**Context and Holism.** Two major themes run through these specific objections, both of which are inspired by the history of the phenomenological movement and the later work of Wittgenstein. One is the global nature of human understanding and the crucial role played by context in comprehension. Facts are not understood in isolation but always as parts of a large-scale structure of meaning. To grasp the significance of any given event, problem, or assertion, one must be able to appreciate its place within a larger context of meanings. Using a notion employed by Husserl (27), the founder of the phenomenological movement, every object and event is perceived within an "outer horizon." It is the outer horizon that, although not itself explicitly perceived, structures and organizes that of which one is aware. In chess it is the human player's overall understanding of the game or board position that implicitly provides the outer horizon within which he perceives a

given piece or move. The second major theme is the claim that the implicit organizing background cannot be made explicit. In particular, it cannot be articulated by a system of relations between context-free elements or as an exhaustive set of determinate rules. This view derives not from the work of Husserl but from that of the later phenomenologists Heidegger (28) and Merleau-Ponty (29) as well as Wittgenstein's *Philosophical Investigations* (30). Husserl, who set out to make the organizing framework of meaning explicit, found the project incomplete as the outer horizon of meaning always receded at his approach. Heidegger went further and argued that Husserl had failed to complete his program, not because he had undertaken an enormous or infinite task, but because the very conception of the project was mistaken. Heidegger stressed that the surrounding context of meaning did not consist just of further beliefs, expectations, and rules. Rather it included the context of social and cultural practices, physical artifacts, tools, and equipment as well as the physical, biological, and historical situation within which human beings live, perceive, and use language. According to Heidegger, it is the actual situations within which human beings live that provide the background for the structure of human meaning, and these situations are not to be confused with a set of beliefs about one's situation or an internalized representation of the situation. It is the situations themselves that provide the boundaries, the limits, and the organizing structure for meaningful behavior. On this view it is simply mistaken to suppose that all of this structure must somehow be internalized in the mind of the agent in order for his action to fit within and derive its significance from the larger context.

This second theme is related as well to Wittgenstein's claim that any attempt to analyze meaningful behavior as acting in accordance with a rule must always require sooner or later the existence of a meaning-giving context that is not itself to be explicated in terms of rule following [30]. Wittgenstein argued that a behavior could be counted as following a given rule only relative to a context. The physical actions of pointing and uttering a sound count as actions of demonstrative reference and naming only relative to a social context that connects these behaviors with a variety of others. And using a word according to a rule to refer on each occasion to the same sort of thing can only make sense relative to a context that determines what is to count as the same kind of thing. If this context were itself to be analyzed in terms of rule following, it would in turn presuppose yet a further context within which the social behaviors would count as rule following. If an infinite and vicious regress is to be avoided, at some point there must be a meaning giving social context that is not itself a matter of following rules.

**Critique of More Recent AI.** It is these two major claims about the global role of context and the impossibility of making the context fully explicit as a system of beliefs or rules that constitute the basis for the continuing phenomenological critique of AI. Given the rapidly changing nature of AI research, Dreyfus's criticisms of early AI programs would be only of historical interest. Indeed, more recent work in AI has sought to remedy many of the defects Dreyfus noted. Chess-playing programs employ descriptions of larger organizational patterns, and language-comprehension programs include a large store of information about the nonlinguistic world. AI programmers have become especially aware of the need to include a great deal of background information about ordinary commonsense matters in their programs, as in Minsky's frames (see Frame theory) (31) or Schank's scripts (qv) (32). But phe-



nomenological critics like Dreyfus argue that though these more recent attempts are improvements over early AI programs, they are certain nonetheless to fall short of achieving genuine intelligence or understanding. These critics see the attempt to program the background context of everyday knowledge as a repetition of Husserl's unsuccessful program of phenomenology and fated to fail for the same reasons. Current AI researchers could be said to have responded to one of the two themes of the phenomenological critique—the importance of context—but to have not accepted the other theme—the claim that the global context should not be thought of as a structure of beliefs, rules, or representations but rather as an actual situation or form of life within which the understander lives. To accept this second claim would be to abandon the project of AI, at least in anything like its current form, and so it is not likely to be acknowledged as was the role of context. As the phenomenological critics admit, the claim that meaning and understanding presuppose a nonrepresentational context is not the sort of claim that can be proved by demonstrative argument (33). Its force derives rather from an overall picture of meaningful human behavior. In the absence of conclusive argument, AI researchers are not likely to abandon their own competing picture, which treats the background or context of meaning as capable of explicit and determinate formalization. The conflict between AI and its phenomenological critics will have to be settled on the basis of AI's subsequent success or failure rather than on the basis of *a priori* arguments.

**Logical Empiricism and AI.** Many of the problems faced by AI were also addressed by the philosophical movement known as logical positivism and its successor logical empiricism (34). Since the methodological assumptions of the positivists were much closer to those of present-day workers in AI than were those of the phenomenologists, philosophers sympathetic to the positivist program are more likely than phenomenologists to be optimistic about the prospects of AI. The philosopher of science, Glymour has argued that AI, or at least those areas of AI concerned with machine learning, should be viewed as continuous with the logical empiricist philosophy of science in their goals and approach despite their minimal direct historical connection (1). The positivist and logical empiricist programs aimed at a rational reconstruction of scientific method. The result was meant to be descriptive in making explicit the methods and reasoning that underlay the success of modern science but was also intended to provide an idealized account of scientific method, which might diverge from and improve on some of the practices of actual scientists. Glymour has pointed out that the goals of the positivist program were similar in many respects to current work in AI (1). The positivists aimed to formulate precise rules to specify such central scientific notions as the empirical or observational content of a theory, the degree of confirmation of a hypothesis by a body of evidence, and the explanation of an event by the appeal to scientific laws. In each case the positivists demanded a high degree of specificity in any rules that were to count as solutions to these or other problems in their program. As Glymour has stressed, it is this demand for specificity as much as the problems addressed that marks the similarity between logical positivism and AI. The AI demand for precise and fully determinate rules, which has drawn criticism from phenomenologists, is what most closely links AI with positivism. The positivist's demand grew out of their epistemological commitment to a

strong form of foundationalism; semantic properties were attributed only to a limited class of statements concerning the simple immediate objects of sense experience, sense data. All other meaningful statements were to be constructed in terms of logical relations defined over sense-data statements. In consequence, the method allowed for virtually no undefined semantic relations. One could not take as understood such notions as being a positive instance of a hypothesis or being an observation consequence of a theory. All such relations had to be spelled out by precise and largely syntactic rules. Even ordinary commonsense notions such as being a physical object or remaining rigid in motion had to be defined precisely by reference to the restricted class of sense-data statements. AI's demand for specificity has a different source. AI rules must be computable if they are to be implemented, and they must not rely on undefined semantic notions since the machine has no prior knowledge of what has not been programmed into it. Thus, Glymour argues, the different methodological constraints of AI and logical positivism impose a common demand for spelling out crucial semantic notions by specific precise syntactic rules. The formal tools employed also differ since logical positivists relied primarily on the predicate calculus, but the goal of formalization remains the same.

**Logical Empiricism and Machine Learning.** The fact that logical positivism is generally regarded as having failed to achieve its goals, need not, Glymour contends, reflect unfavorably on AI. The weakness in the positivistic program lay in its commitment to a sense data foundationalism, which is not shared by AI. One can hope that positivist goals with respect to such problems as formalizing the process of hypothesis confirmation will still be solved by AI. Glymour argues that greater communication between those working on machine learning and those who wish to continue the logical empiricist program would be mutually beneficial. He notes that the AI attempt by Shapiro (35) to model a logic of confirmation on Popper's logic of scientific discovery (36) makes clear just what needs to be done to employ Popper's method of conjecture and refutation. In particular, machine implementation requires specific rules to determine which hypothesis to blame or refute when a negative result is obtained with respect to the prediction entailed by a conjunction of hypotheses. Without solving the problem of assessing blame, the general method cannot be programmed. In the absence of the demands for specificity imposed by machine implementation, the importance of the problem can be overlooked in philosophical discussion. Conversely, as the AI application of Popper's method illustrates, AI has much to gain from research done in the philosophy of science on topics that have naturally and independently arisen in AI.

### Other Areas of Philosophy

Although epistemology and the philosophy of mind are the two subfields of philosophy with the most direct connection to AI, other areas of philosophy such as metaphysics, the philosophy of action, and the philosophy of language also have relevance to AI. As McCarthy and Hayes have noted (37), this is especially true when one wishes to program general understanding, which requires building into the program a great deal of background knowledge about such common but metaphysically central notions as probability, causality, action, intention, and personhood. Such concepts have long been the subjects of intense philosophical investigation within diverse philosophical traditions and with varying degrees of formal

rigor. Though no general philosophical consensus has developed on most of these issues, progress has made and many promising suggestions, which might tempt uninformed workers in AI, have been explored by philosophers and found unacceptable. Among the many particular topics on which the philosophical literature might be of interest to AI researchers would be causation (see Reasoning, causal) (38), counterfactuals (39), conditionals (40), practical reasoning (41), intentional action (42), the structure of events (43), speech acts (qv) (44), and conversational implications (45).

## BIBLIOGRAPHY

1. C. Glymour, Android epistemology: Reflections on Artificial Intelligence and the Philosophy of Science, presented at *Pacific Division Meeting of the American Philosophical Association*, San Francisco, CA, March 1985.
2. A. Turing, "Computing machinery and intelligence," *Mind* **59**, 433–460 (1950).
3. K. Gunderson, "The imitation game," *Mind* **73**, 234–245 (1964).
4. N. Block, "Psychologism and behaviorism," *Philos. Rev.* **90**, 5–43 (1981).
5. P. M. Churchland and P. S. Churchland, "Functionalism, qualia, and intentionality" *Philos. Top.* **12**, 121–145 (1981).
6. M. Scriven, "The mechanical concept of mind," *Mind* **62**, 230–240 (1953).
7. N. Block, What is Functionalism? in N. Block (ed.), *Readings in the Philosophy of Psychology*, Vol. 1, Harvard University Press, Cambridge, MA, pp. 171–184, 1980.
8. D. C. Dennett, *Brainstorms*, MIT Press, Cambridge, MA, 1978.
9. J. Fodor, *Psychological Explanation*, Random House, New York, 1968.
10. H. Putnam, The Nature of Mental States, in W. H. Capitan and D. D. Merrill (eds.), *Art, Mind, and Religion*, University of Pittsburgh Press, Pittsburgh, PA, pp. 37–48, 1967.
11. D. Lewis, "Psychophysical and theoretical identification," *Austral. J. Philos.* **50**, 249–258 (1972).
12. N. Block, Troubles with Functionalism, in C. W. Savage (ed.), *Perception and Cognition. Issues in the Foundations of Psychology Minnesota Studies in the Philosophy of Science*, Vol. 9, University of Minnesota Press, Minneapolis, MN, pp. 261–325, 1978.
13. J. J. C. Smart, "Sensations are brain processes," *Philos. Rev.* **68**, 141–156 (1959).
14. D. C. Dennett, "Intentional systems" *J. Philos.* **68**, 87–106 (1971). Reprinted in Ref. 8.
15. A. Newell, Intellectual Issues in the History of Artificial Intelligence, in F. Machlup and U. Mansfield (eds.), *The Study of Information: Interdisciplinary Messages*, Wiley, New York, pp. 187–228, 1983.
16. S. Stich, "Dennett on intentional systems," *Philos. Top.* **12**, 39–62 (1981).
17. D. C. Dennett, "Why the law of effect won't go away," *J. Theor. Soc. Behav.* **2**, 169–187 (1975). Reprinted in Ref. 8.
18. J. Fodor, "Methodological solipsism considered as a research strategy for cognitive science," *Behav. Brain Sci.* **3**, 63–109 (1980).
19. J. Searle, "Minds, brains and programs," *Behav. Brain Sci.* **3**, 417–457 (1980).
20. J. R. Lucas, "Minds, machines and Gödel," *Philosophy* **36** 112–127 (1961).
21. Reference 8, pp. 256–266.
22. P. Benacerraf, "God, the devil, and Gödel," *Monist* **51**, 9–32 (1967).
23. D. Hofstadter, *Gödel, Escher, Bach: An Eternal Golden Braid*, Basic Books, New York, 1979.
24. E. Gettier, "Is justified true belief knowledge," *Analysis* **23**, 121–123 (1963).
25. H. Dreyfus, *What Computers Can't Do*, 2nd ed., Harper Row, New York, 1979.
26. P. Winston and the staff of the MIT AI Laboratory, Proposal to ARPA, MIT AI Memo 336, Cambridge, MA, 1976.
27. E. Husserl, *Ideas General Introduction to Pure Phenomenology*, MacMillan, New York, 1931.
28. M. H. Heidegger, in J. Macquarrie and E. Robinson *Being and Time*, (trans.), Harper & Row, New York, 1962.
29. M. Merleau-Ponty, *Phenomenology of Perception*, Routledge and Kegan Paul, London, 1962.
30. L. Wittgenstein, *Philosophical Investigation*, Blackwell, Oxford, 1953.
31. M. Minsky, A Framework for Representing Knowledge, Memo 306 MIT Artificial Intelligence Laboratory, Cambridge, MA. Excerpts published in P. H. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, pp. 211–277, 1975.
32. R. Schank and R. Abelson, *Scripts, Plans, Goals and Understanding*, Lawrence Earlbaum, Hillsdale, NJ, 1977.
33. D. Hofstadter, *Gödel, Escher, Bach: An Eternal Golden Braid*, 2nd ed., Basic Books, New York, 1979.
34. A. J. Ayer (ed.), *Logical Positivism*, MacMillan, New York, 1959.
35. E. Shapiro, *Algorithmic Program Debugging*, MIT Press, Cambridge, MA, 1983.
36. K. R. Popper, *The Logic of Scientific Discovery*, Hutchinson, London, 1959.
37. J. McCarthy and P. J. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 4, Halstead, New York, pp. 463–502, 1969.
38. E. Sosa, *Causation and Conditionals*, Oxford University Press, London, 1975.
39. D. Lewis, *Counterfactuals*, Harvard University Press, Cambridge, MA, 1973.
40. W. Harper, *Ifs: Conditionals, Belief, Decision, Chance and Time*, Reidel, Dordrecht, The Netherlands, 1980.
41. D. Jeffrey, *The Logic of Decision*, McGraw-Hill, New York, 1965.
42. A. Goldman, *A Theory of Human Action*, Prentice-Hall, Englewood Cliffs, NJ, 1970.
43. D. Davidson, *Essays in Actions and Events*, Oxford University Press, New York, 1980.
44. J. Searle, *Speech Acts*, Cambridge University Press, Cambridge, U.K., 1969.
45. H. P. Grice, Logic and Conversation, in D. Davidson and G. Harman (eds.), *The Logic of Grammar*, Dickenson, Encino, CA, pp. 64–74, 1975.

## General References

- A. R. Anderson, *Minds and Machines*, Prentice-Hall, Englewood Cliffs, NJ, 1964. An anthology of important articles from the early period of the mind-machine debate.
- P. M. Churchland, *Matter and Consciousness: A Contemporary Introduction to the Philosophy of Mind*, MIT Press, Cambridge, MA, 1984.
- H. Dreyfus (ed.), *Husserl, Intentionality and Cognitive Science*, MIT Press, Cambridge, MA, 1982. A collection of essays bringing the phenomenological perspective to bear on work in AI and cognitive science.
- C. Glymour, *Theory and Evidence*, Princeton University Press, Princeton, NJ, 1980. A recent detailed account of confirmation.



- J. Haugeland, *Mind Design Philosophy, Psychology Artificial Intelligence*, MIT Press, Cambridge, MA, 1982. An excellent anthology of recent philosophical and theoretical work on AI.
- J. Lucas, *The Freedom of the Will*, Oxford University Press, Oxford, U.K., 1970. Includes further discussion of the Gödel argument with bibliography.
- F. Suppe (ed.), *The structure of Scientific Theories*, 2nd ed., University of Illinois Press, Urbana, IL, 1979. Includes a critical introduction that provides an overview of recent work in the philosophy of science that would be of relevance to AI.

R. Van GULICK  
Syracuse University

## PHONEMES

Perhaps the most remarkable thing about the ability to speak a language is that it is based on an illusion (see Speech acts). When one listens to someone who is speaking, the hearer believes that he or she is hearing strings of words being said one after the other and that these words are separated from one another just as the words in this paragraph are separated by spaces. But this is patently not the case. Consider, for example, the following phrases:

1. Mice lick ice.
2. My slick ice.

These two phrases sound exactly alike when they are uttered in most dialects of American English. Moreover, were a sensitive recording device such as a sound spectrograph to record most English speakers saying sentence 1 and then 2, there would be no way of telling which utterance was being said. And yet any speaker of English is able to tell that the single speech stream that emanates from a reading of either 1 or 2 corresponds to two separate strings of words in English. This indicates that words are not concretely delimited in the stream of speech and therefore that our knowledge of words must be to a considerable extent abstract. In other words, speakers can tell where words begin and end even though there is no physical marker in the speech stream that carries that information. Hence, where words begin and end must be abstract knowledge.

### Phonemes

Part of one's abstract knowledge of English words is that they are themselves made up entities called phonemes. Phonemes are, roughly speaking, the building blocks from which words are constructed. Thus, everyone knows that words are made up of sounds, and the sounds that make them up are called phonemes. It has already been suggested that there are no boundaries between the words that follow one another in a spoken sentence. Similarly, there are no pauses between the phonemes that make up a given word. Just as a sound spectrogram will show this for strings of words, it will also do the same for the strings of phonemes that make up words. Nonetheless, speakers are able to divide words up into their phonemic parts even though no spaces exist between those parts.

### Rhymes

Consider Alexander Pope's *Epigram: Engraved on the Collar of a Dog which I gave to his Royal Highness*:

*I am his Highness' Dog at Kew;  
Pray tell me Sir, whose Dog are you?*

It is clear that the words at the end of each line in this couplet rhyme. What does it mean for words to rhyme in English? Two words rhyme if two conditions are met:

1. The two words are identical with respect to the string of sounds beginning with the stressed vowel and ending with the end of the word. Call this string the same string.
2. The two words differ with respect to the string of sounds from the beginning of the word up to the stressed vowel. Call this string the different string.

The rhyming words in the Pope couplet are these:

Different String	Same String
K	ew
Y	ou

Other more complicated examples of rhyming words are these:

Different String	Same String
B	ucket (bucket)
Nant	ucket (Nantucket)
Sal	ami (salami)
Pastr	ami (pastrami)
Macar	oni (macaroni)
P	ony (pony)

Notice that in order to tell whether two words rhyme or not, it must be possible for a speaker of English to divide those words into smaller parts. In particular, it must be possible to find that boundary that separates the different string from the same string. This ability to decompose words must be based on abstract knowledge since there are no markers within a word that separate these parts.

### Plural Rule

Another example of the ability to find the parts in words even though they are not physically marked in the string is the rule whereby speakers select the appropriate plural ending for words. Consider the following words:

I	II	III
church	cap	cab
judge	cat	cad
kiss	cake	cog

The words in column I all take a plural ending composed of the syllable [ɪz]. The words in column II all take a plural ending [s], whereas the words in column III take the plural ending [z]:

I	II	III
church + ɪz	cap + s	cab + z
judge + ɪz	cat + s	cad + z
kiss + ɪz	cake + s	cog + z

English speakers must be able to divide words up into phonemes in order to be able to associate the correct plural ending

with each word type. In particular, they must be able to separate the last phoneme in a word from the rest of the word and then select the appropriate plural in terms of that last phoneme.

It is worth considering how English speakers assign the right plural ending to a given word. One way that one might perform this task is by making a list so that the plural rule in English looks as follows:

Assign the plural [ɪz] to a noun if it ends in one of the following phonemes: [s z sh zh ch dj].

Assign the plural [s] to a noun if it ends in one of the following phonemes: [p t k f TH].

Assign the plural [z] to a noun if it ends in any phoneme not on the preceding two lists.

Where TH stands for the final sound in *teeth*  
th stands for the final sound in *bathe*  
sh stands for the final sound in *bush*  
zh stands for the final sound in *rouge*  
ch stands for the final sound in *church*  
dj stands for the final sound in *judge*

However, there is another possibility: Phonemes are not indivisible entities as suggested by the plural rule, which simply lists phonemes, but rather, like atoms, composed of various subatomic particles, phonemes are collections of subphonemic properties. These properties are called distinctive features.

Consider what phonemes might look like according to the view that they are, in fact, not indivisible entities but rather are bundles of distinctive features. To begin with, English contains the following consonants:

[p t k b d g f v TH th m n s z sh zh ch dj]

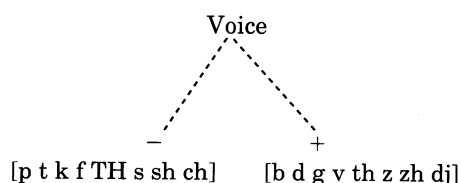
These consonants can be divided into two separate classes, those pronounced without the vocal cords vibrating, the so-called voiceless consonants,

[p t k f TH s sh ch]

and those pronounced with the vocal cords vibrating, the so-called voiced consonants:

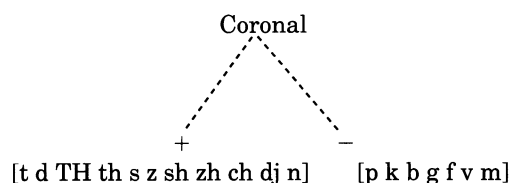
[b d g v th z zh dj]

It is fairly easy to tell which is which. If one places one's thumb and forefinger gently on the adam's apple and pronounces first b and then p, the voiced consonant b will produce a vibration that will be felt in the fingertips, whereas the voiceless consonant will not. This vibration is caused by the vocal cords vibrating. One can represent the division of English consonants into voiced and voiceless sets in the following way:



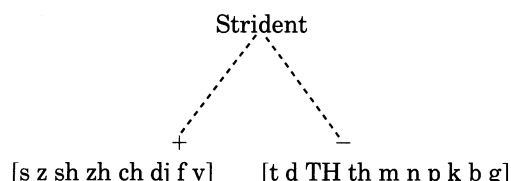
Thus, one would say that with respect to the distinctive feature [voice], the consonants on the left of the above tree are [-voice] and those on the right are [+voice].

English consonants can be divided along another dimension, namely, whether the tip of the tongue is used in producing the sound. This dimension is called coronal, and the consonants divide up as follows:



The tip of the tongue is engaged in pronouncing the first consonant in *tip* while it is not engaged in the first consonant in *pit*. Those on the left are [+coronal] and those on the right are [-coronal].

Another dimension along which consonants can be divided is stridency. Sounds that are strident are those produced with a hissing and/or hushing quality as in the last consonant in *rouge* and the first consonant in words like *sick*, *Zack*, *Shick*, *chick*, *Jack*, *fat*, *vat*:



Those on the left are [+strident] and those on the right are [-strident].

In terms of distinctive features the plural rule would look like this:

Assign the plural ending [ɪz] to a noun if it ends in a phoneme whose distinctive feature bundle contains

[+strident], +coronal]

If the phoneme does not contain these features, assign the plural ending [s] to the noun in question if it ends in a phoneme whose distinctive feature bundle contains

[-voice].

If the phoneme does not contain the feature [-voice], assign the plural ending [z].

### How the Distinctive-Feature-Based Plural Rule Works

Consider how this rule works. Given a noun in English that takes a regular plural (excluding irregular plurals like *children* from *child*, *oxen* from *ox*, or *mice* from *mouse*), a speaker of English must be able to separate the last phoneme of a word from all of the others. Then, on the assumption that phonemes are, in fact, bundles of distinctive features, the speaker must be able to examine the bundle of features that makes up the last phoneme in the noun and determine whether the feature bundle contains the features [+strident, +coronal]. If it does, the assignment of the plural ending [ɪz] is made. If the feature bundle does not contain those two features, the next rule applies. Once again, having separated the last phoneme from all those that precede in the word, the speaker examines the feature bundle to see if the feature [-voice] is present. If it is, the plural ending [s] is assigned. If this feature is not present, the speaker applies the third rule, which assigns the plural ending

[z] to all words that have failed to acquire a plural ending through the prior operation of the first two rules.

### How The Two Plural-Rule Systems Compare

The reader may feel that both rule systems are equivalent since they both yield the same correct results. It is easy to show that this is not the case. Consider a novel form, say *speX*, where the final sound [X] is meant to indicate the sound that appears in the German pronunciation of the name *Bach*. What claims will each theory make about the plural of this novel word? The plural rule as a list will predict that the plural of this form ought to end in [z] since the sound [X] does not appear on any of the lists in the first two rules and therefore the third (default) rule will apply. However, the plural rule as a distinctive-feature rule predicts that the plural ending for this novel form is [s]. Since the phoneme [X] is [-coronal, -voice], the first distinctive feature rule will not apply, but the second one will, yielding the [s]. A moment's reflection will show that the latter result is, in fact, the correct one. That is, the only possible plural ending for a novel word like *speX* is [s] and not [z]. It does not matter that this word is not a word in English. What matters is that an English speaker, when confronted with a word like this, will always assign a plural ending in terms of the plural rule as a distinctive-feature rule. This strongly suggests that speakers of English have acquired something very much like the distinctive-feature-based plural rule of English.

### Conclusion

Although this discussion has only focused on a single rule of English, nevertheless one can make some rather important observations about English phonology and phonological systems in general. The first is that words must be represented as composed of sequences of phonemes. The second is that phonemes must be represented as bundles of distinctive features such as [strident], [coronal], and [voice], to name just three of the 20-odd distinctive features that many phonologists now assume operate in the languages of the world (although never all 20 at once in a given language).

However, perhaps the most important observation of all concerns how human beings are constructed to perceive external stimuli. At least from the linguistic perspective, it seems clear that although the input that speakers hear as language is, in fact, a continuous stream of sounds, nonetheless speakers are endowed with a mental apparatus that forces them to treat that continuous speech stream as segmental in character. Indeed, speakers can no more force themselves to hear the continuous nature of the speech stream than they can force themselves to see the radio sky with the naked eye. The essential function of phonology is just that: to make the continuous discrete.

### General References

- G. N. Clements and S. J. Keyser, *CV Phonology: A Generative Theory of the Syllable*, Linguistic Inquiry Monograph Series, No. 9, MIT Press, Cambridge, MA, 1983.
- F. Dell, *Generative Phonology*, Cambridge University Press, London, 1980.
- M. Halle and G. N. Clements, *Problem Book in Phonology*, MIT Press, Cambridge, MA, 1983.
- M. Kenstowicz and C. Kisseberth, *Generative Phonology: Description and Theory*, Academic Press, NY, 1979.
- S. J. Keyser, Rules and Principles in Phonology and Syntax, in F. Machlup (ed.), *The Study of Information: Interdisciplinary Messages*, Wiley, New York, 1983.
- S. J. Keyser, Why Study Human Language? in M. Gazzaniga (ed.), *Cognitive Neurosciences*, Plenum, New York, 1983.
- S. J. Keyser and W. O'Neil, *Rule Generalization and Optionality in Language Change*, Studies in Generative Grammar 23, Foris, Dordrecht, 1985.
- P. Ladefoged, *A Course in Phonetics*, second edition, Harcourt Brace Jovanovich, NY, 1982.
- K. N. Stevens, S. J. Keyser, and H. Kawasaki, Toward a Phonetic and Phonological Theory of Redundant Features, with Kenneth N. Stevens and Haruko Kawasaki in J. Perkell and D. H. Klatt (eds.), *Studies in Invariance*, Lawrence Erlbaum, Hillsdale, NJ, 1985.

S. J. KEYSER  
MIT

### PHRAN AND PHRED

A natural-language analyzer and a natural-language generator (see Natural-language generation) that have been developed around 1980 and used at the University of California at Berkeley in Robert Wilensky's research group, these two programs have served as the front-end and back-end, respectively, of planning systems like PAM (qv) and UC. PHRAN was written by Yigal Arens, and PHRED was written by Steve Upstill and Paul Jacobs (see R. Wilensky, *Planning and Understanding*, Addison-Wesley, Reading, MA, 1983, and F. Rose, *Into the Heart of the Mind*, Harper & Row, New York, pp. 98-115, 1984).

K. S. ARORA  
SUNY at Buffalo

### PHYSICS, NAIVE

Naive physics is the body of knowledge that people have about the surrounding physical world. The main enterprise of naive physics is explaining, describing, and predicting changes in the physical world. There is an important distinction between classical physics and naive physics. Classical physics is based on the presupposition that there is a shared unstated common sense prephysical knowledge rooted in experience. Naive physics is this prephysical knowledge rooted in experience. It is important to notice that in classical physics, concepts like state, law, cause equilibrium, oscillation, momentum, feedback, etc., are qualitative in nature. However, they have been embedded in a complex framework established by the mathematics of real numbers and differential equations. The relationship between qualitative (common sense) models and mathematical models can be stated as shown in Figure 1. Qualitative simulation captures less detail and therefore may produce partial behavioral descriptions. Also, the quantitative precision of these descriptions is reduced while crucial distinctions are retained.

A research directed at understanding and modeling naive-physics reasoning concentrates mainly on deriving the qualitative concepts used in naive physics from formal models and identifying the core knowledge underlying physical intuition

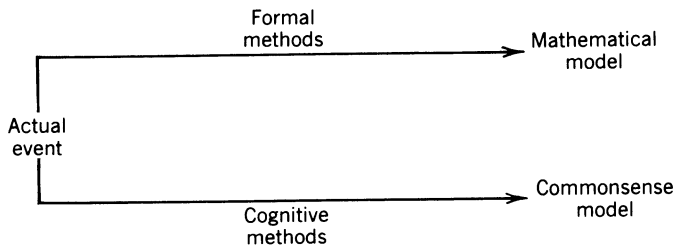


Figure 1

(1). The main conjectures guiding the research are that naive-physics knowledge, to a large degree, can be derived from commonsense observation and that quantitative laws can be mapped to qualitative ones.

Reasoning (qv) about physical changes is the main motivation behind acquiring naive-physics knowledge. Since, by their nature, these changes involve physical (dynamic) systems that undergo some transformation over time, they can be viewed from two perspectives: a device-centered perspective and a process-centered perspective. Each perspective generates its own conceptual vocabulary and its own set of relations between concepts. A brief review of current research on the representation of naive-physics knowledge from the above two perspectives is given in the following. In addition, a brief discussion on the issue of the organization of knowledge in mental models and their cognitive validity will follow. A detailed state-of-the-art account of the research on naive physics can be found in recently edited collections of papers (2-4).

In the device-centered ontology developed by De Kleer and Brown (5,6) the function of a dynamic system is derived from its structure (see Qualitative physics). However, it is of crucial importance that in the knowledge representation used to model the reasoning, the function must not be encoded in the structure. The main concern here is the general form of the physical law and the calculi for deriving the inferences from this law. Causality gets central treatment—how the systems achieve their behavior is as important as what the behavior is. Consequently, in this ontology the world is viewed as a complex machine with interacting components. The laws for a component of a dynamic system do not influence other parts. Classwide assumptions are made for the generic systems distinguishing them from idiosyncratic assumptions for a particular system. The approach to modeling is reductionist, and the types of physical constituents considered are materials, components, and conduits. Graph topology is used for modeling devices with nodes as components and edges as conduits. The laws in the system must be time-invariant. Laws describing behavior over time require explicit integrals in their formulation. Relations are stated as confluences that are qualitative differential equations, where time is made discrete by states, and the physical parameter values are made discrete by a mapping to quantity space. The states determine the operating range of the confluences. The program ENVISION, which models some aspects of the theory, takes as input a set of components and their allowable paths of interaction, an input signal to the system, and a set of boundary conditions that constrain the system. Since qualitiveness may sometimes underdetermine behavior, the output of the program consists of a set of possible behaviors of the system and not a single unique solution.

The qualitative process theory developed by Forbus, (7,8) is a process-centered ontology and is not equational in nature.

The main enterprise of the theory is reasoning about processes, their effects, and their limits. Quantity space separates magnitudes of equations from their signs:  $[-1, 0, 1]$ . Here, changes in physical situation perceived as causal due to one's interpretation of them are corresponding either to direct changes caused by processes or propagation of those direct effects through functional dependencies. Also, histories, which consist of episodes, are used as descriptions of objects that are extended through time but are always spatially bound.

Kuipers (9) makes the jump from differential equations directly to qualitative constraints among state variables of a system (structural description  $\rightarrow$  behavioral description  $\rightarrow$  functional description). The components in this framework are only state variables, and the connections are the constraints. This simulation assumes that causality is identical to value propagation with constraints.

Gentner (10) observes that the analogical models used in science can be characterized as structure mapping between complex systems. Such an analogy conveys that similar relational systems hold within two different domains. The predicates of the base domain (the known domain)—particularly the relations that hold among the objects—can be applied in the target domain. Thus, a structure-mapping analogy asserts that identical operations and relationships hold among non-identical things. The relational structure is preserved but not the objects.

It seems that naive-physics reasoning has two characteristics that make its scientific inquiry very hard: size—very large amounts of knowledge are involved—and specialized structures—the knowledge structures that are apparently used by humans for naive-physics reasoning are a product of many years of experience interacting with the physical world and hence are “compiled” to a great extent. This makes it very hard to extract their content.

Naive physics is not bad physics. It is different from classical physics both in scope and in power, and its value is measured by its practical validity. For example, consider an everyday life event in which at supper the soup is too hot and one is required to estimate how long it would take the soup to reach a comfortable temperature. Classical physics, given the current knowledge in fluid mechanics and transport processes and the current computer technology, is unable to give a meaningful answer within the natural time constraints imposed by the situation. However, naive physics has a practical solution and almost anyone can use common sense to figure out what to do.

Undoubtedly, naive-physics knowledge about the dynamic and the static nature of the everyday physical world is vital to any agent who plans to act successfully. Furthermore, understanding the way this knowledge is acquired and modified through use can shed light on the nature of human information processing.

## BIBLIOGRAPHY

1. J. R. Hobbs and R. C. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, 1985.
2. D. G. Bobrow and P. J. Hayes (eds.), *Artif. Intell.* (special volume on qualitative reasoning about physical systems) 24(1-3), (1984).
3. D. Gentner and A. L. Stevens (eds.), *Mental Models*, Lawrence Erlbaum, Hillsdale, NJ, 1983.
4. P. Hayes, in ref. 1, “The Naive Physics Manifesto,” pp. 1-36.

5. J. de Kleer and J. S. Brown, in Ref. 2, "A Qualitative Physics Based on Confluences," pp. 7-83.
6. J. de Kleer and J. S. Brown, in Ref. 3, "Assumptions and Ambiguities in Mechanistic Mental Models," pp. 155-190.
7. K. D. Forbus, in Ref. 2, "Qualitative Process Theory," pp. 87-168.
8. K. D. Forbus, in Ref. 3, "Qualitative Reasoning about Space and Motion," pp. 53-73.
9. B. Kuipers, in Ref. 2, "Commonsense Reasoning about Causality: Deriving Behavior from Structure," pp. 169-203.
10. D. Gentner, in Ref. 3, "Flowing Waters or Teaming Crowds: Mental Models of Electricity," pp. 99-129.

S. L. HARDT  
SUNY at Buffalo

## PLANES

A natural-language front-end in ATN, PLANES was developed by Waltz at the University of Illinois in 1978 to work on a relational data base (see Grammar, augmented-transition-network). It was designed from an engineering viewpoint, thus allowing a nongrammatical sequence of words as valid input [see D. Waltz, "An English language question answering system for a large relational database," *CACM* 21(7), 526-539 (1978)].

A. HANYONG YUHAN  
SUNY at Buffalo

## PLANNER

A LISP-based AI programming language for inference control, PLANNER was designed in 1972 at the MIT AI Lab by Hewitt and extensively demonstrated by Winograd in his SHRDLU (qv) project [see G. Sussman, T. Winograd, and E. Charniak, *MICRO-PLANNER Reference Manual*, AI Memo 203, AI Laboratory, MIT, 1970, and C. Hewitt, "Description and Theoretical Analysis (Using Schemata) of PLANNER, a Language for Providing Theorems and Manipulating Models in a Robot," Report No. TR-258, AI Laboratory, MIT, 1972].

A. HANYONG YUHAN  
SUNY at Buffalo

## PLANNING

Planning is the generation of an action sequence or action program for an agent, such as a robot, that can change its environment. The purpose of such a plan is to achieve one or more explicitly stated goals. The essential inputs for planning are an initial world state, a repertoire of actions for changing that world, and a set of goals. The form of the plan is commonly just a linear sequence or acyclic-directed graph, although the full range of programming control structures are potentially relevant. For planning to be effective, the world in which the plan will be executed must be largely predictable, if not completely deterministic. Planning will be ineffective for chaotic domains, and an agent can only react to events. Planning may require a search through an enormous space of possible plans, and so search control is an important consideration.

Other key planning topics are the representation of actions, goal protection, management and modeling of time (see Reasoning, temporal), and the ordering of goals and subgoals for achievement. Areas of application for planning include automated manufacturing (see Computer-integrated manufacturing), robotics (qv), autonomous vehicles (qv), and control of unmanned spacecraft.

## The Blocks World

Many important principles of planning can be illustrated on a simple model called the blocks world. This is a two-dimensional world consisting of a table of unbounded size and a number of square blocks labeled with distinct letters that can be arranged to form stacks. Figure 1 shows a typical blocks-world-planning problem. Blocks-world states are described by four kinds of literals:

- |                |  |
|----------------|--|
| (CLEAR $x$ )   | There is no block on top of $x$ (this representation became common because early planners were unable to handle negations and existential quantifiers) |
| (HOLDING $x$ ) | A hand of an agent is holding block $x$  |
| (ON $x$ $y$ )  | Block $x$ rests directly on block $y$  |
| (ONTABLE $x$ ) | Block $x$ rests directly on the table  |

In general, initial-state descriptions for planners consist of a conjunction of assertions (literals). With these conventions the initial state in Figure 1 is represented by the following conjunction: (CLEAR C) (ON C A) (ONTABLE A) (CLEAR B) (ONTABLE B).

## Representation of Actions

An action is any change in the world state caused by an agent executing a plan. Actions may conveniently be partitioned into primitive and macro actions. This is a relative distinction and depends on the purposes of the planning. A primitive action is one whose finer details are not of interest. A macro action is an aggregate of primitive actions and other macro actions and is analogous to a subroutine in algorithmic programming languages. For example, for a travel agent planning an itinerary "flying from major-city- $x$  to major-city- $y$ " may be a convenient primitive action; for an airline pilot, on the other hand, this is a large macro action, and the primitives are actions such as "set flaps for takeoff" and "taxi to runway." All of the normal control mechanisms potentially apply in forming macro actions: sequencing, conditionals, concurrency, looping, etc. However, most contemporary planners do not provide for loops or conditionals, and the final plan is commonly a partially ordered network of primitive actions. Macro actions are also known by a variety of synonyms, such as "skeletal plans," "plan schemas," and "scripts."

It is often important to distinguish an event from an action. An event is a spontaneous change in world state. An event is

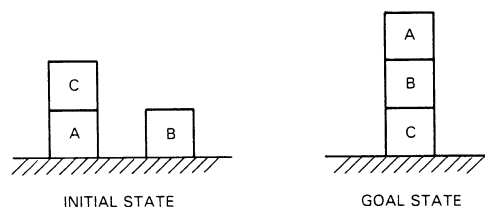


Figure 1. A blocks-world planning problem.

triggered automatically because of natural or artificial mechanisms, whereas an action occurs only if an agent elects to perform it. A person can set a clock radio by moving the alarm control to on. This is an action. The next morning the radio switches on and the person wakes up. These latter two changes are events. If the causes and effects of events are known, it is possible to generate plans in which events play a role in achieving goals. However, the agent cannot directly “execute” the events but can only take actions leading to world states that trigger the desired events.

Primitive actions may be modeled by specifying their preconditions and postconditions, both of which are conjunctions of literals. Preconditions are conditions that must be true in the world for an action to take place. Postconditions are conditions that hold after the action has occurred. Table 1 presents the action definitions for the four blocks-world primitives: PICKUP, PUTDOWN, STACK, and UNSTACK. The definitions are given in the form

```

((action-name) <action-variable-list>
  <preconditions>
  →
  <postconditions>)
```

As defined, PICKUP is the change from a single block resting directly on the table to the state in which a hand is holding the block. Postconditions commonly contradict preconditions, as in this example. Inclusion of (NOT (CLEAR block)) as a postcondition is a somewhat arbitrary, subtle decision that prevents the planner from attempting to stack a block on another one that is being held. Preconditions could be augmented to include negations of positive literals in the postcondition. For example, (NOT (HOLDING block)) could be added to the preconditions of PICKUP. However, this is usually regarded as redundant. This particular formulation of the blocks world permits unbounded concurrency since any number of hands are assumed available for the PICKUP and UNSTACK actions. In these action descriptions the variable name “block” is purely mnemonic and does not formally convey any type information.

Primitive events can be modeled in exactly the same way as actions and may be inserted into plans by the same backward-chaining process (see Processing, bottom-up and top-down) of node expansion as that described below. The causes of an event are listed as preconditions and the effects as postconditions. The difference is that event rules must also be allowed to chain forward whenever their preconditions are accidentally satisfied. The action–event distinction is not mentioned again; in most of the remaining discussions actions can be taken to include events as well.

Logical inferences (see Logic; Inference) can also be written in the same format as action-and-event rules. For example, a global “ABOVE” literal in the blocks world might be inferred with the following rule:

```

(ABOVE INFERENCE
  ((ON block1 block2)
   (ON block2 block3))
  →
  ((ABOVE block1 block3))).
```

**Table 1. The Four Blocks-World Actions**

```

(PICKUP (block)
  ((ONTABLE block)
   (CLEAR block))
  →
  ((HOLDING block)
   (NOT (CLEAR block))
   (NOT (ONTABLE block))))

(PUTDOWN (block)
  ((HOLDING block))
  →
  ((CLEAR block)
   (ONTABLE block)
   (NOT (HOLDING block))))

(STACK (top-block under-block)
  ((CLEAR under-block)
   (HOLDING top-block))
  →
  ((CLEAR top-block)
   (ON top-block under-block)
   (NOT (HOLDING top-block))
   (NOT (CLEAR under-block))))

(UNSTACK (top-block under-block)
  ((ON top-block under-block)
   (CLEAR top-block))
  →
  ((HOLDING top-block)
   (CLEAR under-block)
   (NOT (CLEAR top-block))
   (NOT (ON top-block under-block))))
```

The assertions in the postcondition of an inference rule (in the context of planning) are sometimes called derived assertions. It is permissible to write derived-assertion literals as preconditions in other action, event, or inference rules and to achieve these subgoals by backward chaining with inference rules. However, derived assertions have the unique property that, unlike action-and-event postconditions, they remain true exactly as long as their preconditions remain true. This has implications for the way goal-protection relations are managed for derived assertions and is explained below (after the goal-protection topic has been properly introduced).

It should be emphasized that an action description in precondition–postcondition form is not a logical implication in the first-order predicate calculus (fopc) (see Logic, predicate). However, one early approach to planning did propose the modeling of actions with first-order logic (1,2). This requires the augmentation of domain assertions with state variables. For example, (CLEAR block s1) does not contradict (NOT (CLEAR block s2)) if state variables s1 and s2 are distinct. The putdown action above could then be written as the following first-order axiom:

```

Forall s1 (HOLDING block s1) →
  (CLEAR block (PUTDOWN block s1))
  (ONTABLE block (PUTDOWN block s1))
  (NOT (HOLDING block (PUTDOWN block s1)))
```

where (PUTDOWN block s1) is the new state reached by applying PUTDOWN to s1.

The problem is that a typical action changes only a few of the many assertions comprising a complete world description and says nothing about whether other assertions are still true

in the new state. A separate "frame axiom" is required for each action and assertion type to allow a theorem prover to infer that the rest of the world is unchanged in the new state. For example, the following axiom is required to propagate ON assertions across a PUTDOWN action:

Forall x, y, s, block (ON x y s)  $\rightarrow$   
 (ON x y (PUTDOWN block s))

Frame axioms are tedious to write and a crushing computational load for any theorem prover that must apply them. This "frame problem" was the subject of some concern and debate in the early 1970s. Fortunately, it eventually became clear that this is not an intrinsic impediment to planning in general. The proper conclusion is that feeding first-order action and frame axioms into a general-purpose theorem prover is not an efficient way to do planning. It should be emphasized, however, that the two papers (1,2) made other research contributions of continuing significance.

In the NOAH-NONLIN class planners described below, world states are distributed over a partially ordered network of nodes, and no explicit state updating is necessary. Every literal is identified with a specific node in the network that "asserts" that literal. Ordered nodes may assert contradictory literals. Propagation of assertions across actions is implicit in these planners and in effect relies on two higher-order frame axioms:

Forall literal, node

(ASSERTS node literal)  $\rightarrow$  (TRUE.AT node literal)

Forall literal, node1, node2

(TRUE.AT node1 literal)

(ORDERED node1 node2)

(NOT (ASSERTS node2 (NOT literal)))

$\rightarrow$

(TRUE.AT node2 literal)

### A Planner in Operation

Many of the basic planning issues and procedures can be illustrated by following through the major steps of a planner as it solves the problem of Figure 1. The planner to be illustrated is a slightly generalized version of NONLIN (3). The planner keeps its goals on a stack and does an ordered depth-first search (qv) through the space of possible plans. It synthesizes its plan from the primitive blocks-world actions defined above.

The state of the planner is largely summarized by the plan diagrams shown in Figures 2–8. The final plan is shown in Figure 9. In these diagrams unordered actions are concurrent. Each node in the diagrams, except the stop node, has associated with it a set of assertions. For the start node these assertions are the set of literals describing the initial state. For action nodes the assertions are the instantiated postconditions of the action. The top goal on the planner's goal stack is denoted by a double-outlined box. Figure 2 shows the initial plan diagram. Three goal nodes are shown in parallel; (ONTABLE C) will be attempted first. Not shown, to avoid cluttering the plan diagrams, are the node assertions and the goal protection relations (4) (also called collectively the "goal structure"), which govern the protection of goals within a plan.

Goal protection is described by meta-level assertions (see

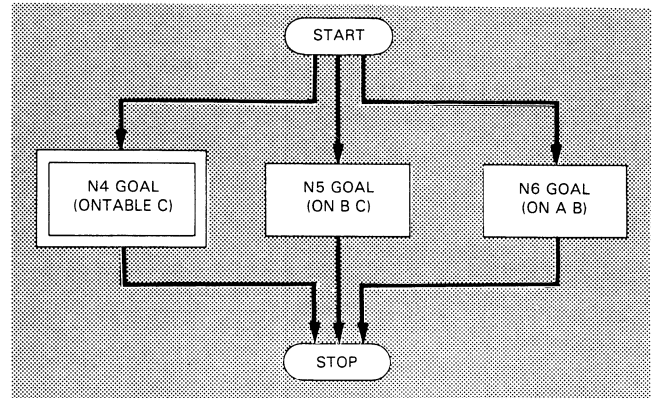


Figure 2. Initial plan diagram for the blocks-world problem.

Meta-knowledge, -rules, and -reasoning) of form (PROTECTED (literal) (node1) (node2)), which signify that (literal) must not be contradicted in the region of the plan between (node1) and (node2) and where (node1) asserts (literal) and (node1) precedes (node2) in the plan diagram. In constructing the initial diagram in Figure 2, the planner records the following goal protections: (PROTECTED (ONTABLE C) N4 STOP) (PROTECTED (ON B C) N5 STOP) (PROTECTED (ON A B) N6 STOP). Protection relations are the sinews that bind together the elements of a plan and ensure its coherence. In this example they ensure that the three goal conditions hold simultaneously when the plan terminates. Without goal protection a planner may merely achieve one goal at the expense of another, like the crow in Aesop's fable who drops the cheese when she opens her mouth to sing.

If a goal is not already true, the planner must take action to make it true. It selects an action with a postcondition literal that matches the goal, and the preconditions of that action become new goals. Placing the new goals on top of the goal stack achieves a depth-first search. In the blocks world, only a PUTDOWN action asserts an ONTABLE literal, and so there

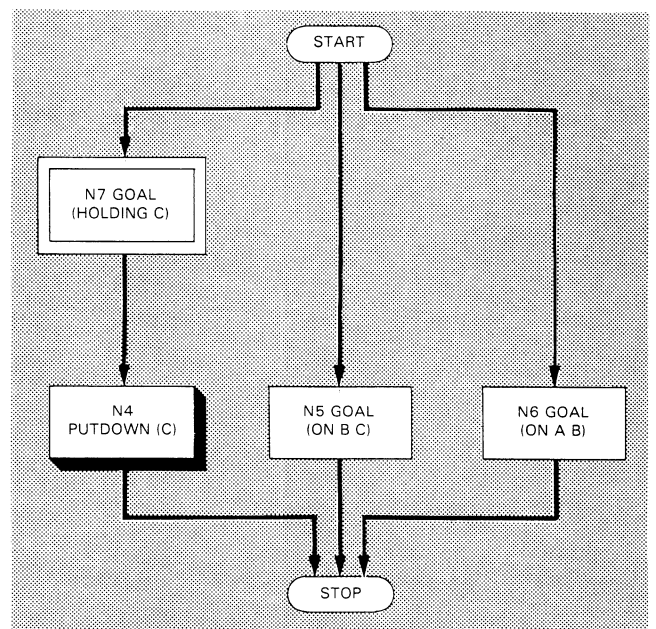


Figure 3. Plan diagram after step 1.



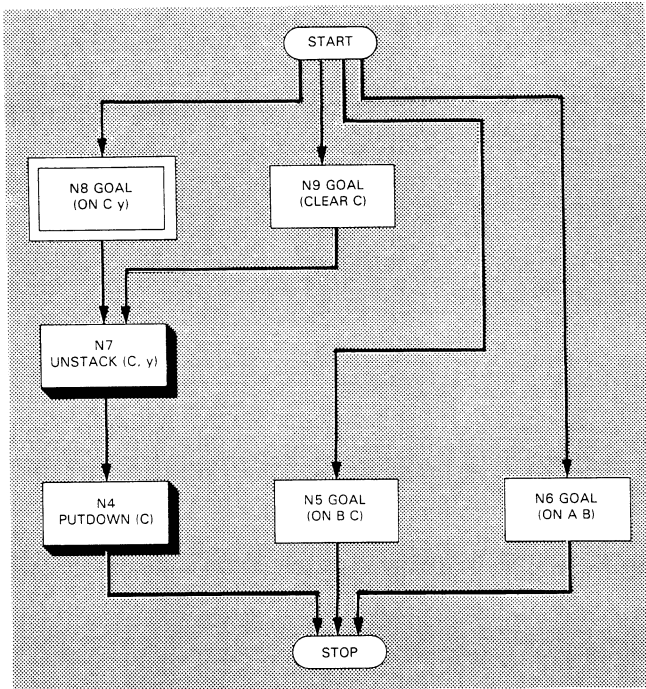


Figure 4. Plan diagram after step 2.

is no choice. This process is called node expansion, and the result is shown in Figure 3. (The generic term for node expansion, backward chaining, applies to a broader range of AI systems than just planners.) Protection relations are established between the new goal node, N7, and the parent node N4. The instantiated postconditions of PUTDOWN become the assertions of N4: (ONTABLE C) (NOT (HOLDING C)) (CLEAR C). If there are several actions that can achieve the goal, this amounts to an OR branch in the search tree. One action must

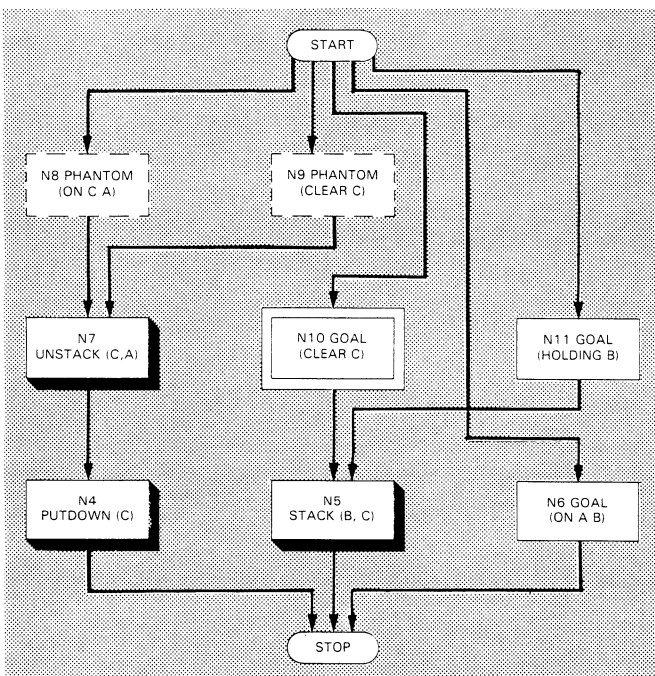


Figure 5. Plan diagram after step 5.

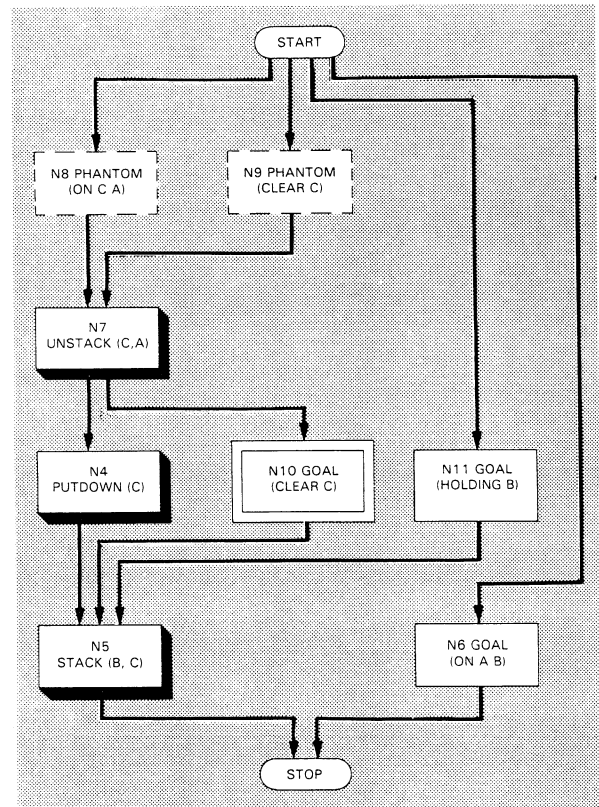


Figure 6. Plan diagram after step 7.

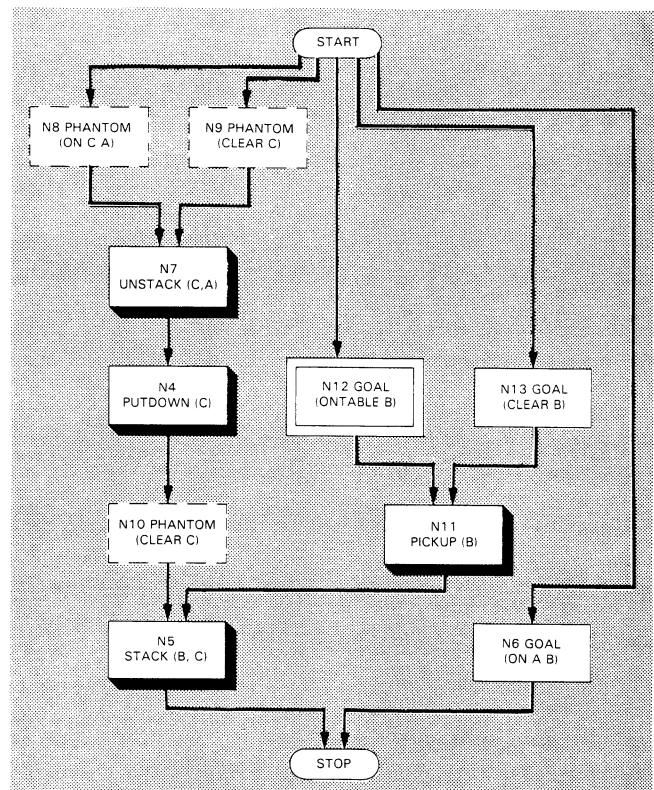


Figure 7. Plan diagram after step 9.

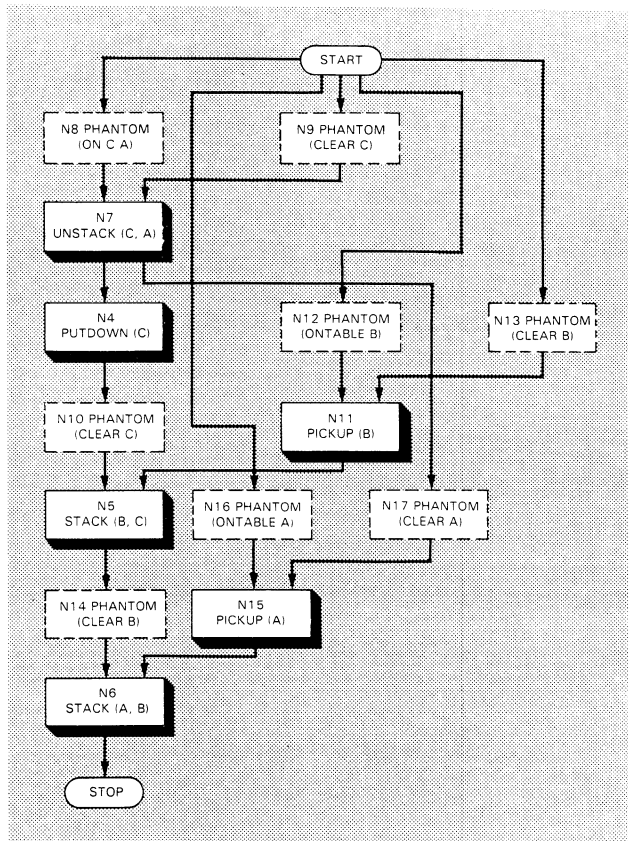


Figure 8. Final plan diagram.

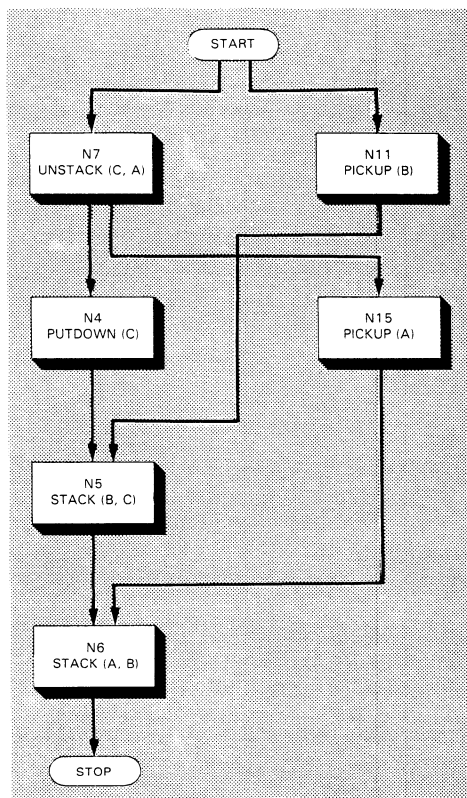


Figure 9. Final blocks-world plan (without phantoms).

be selected for expansion and the others remembered for backtracking (qv). Expanding goal node N7 with an UNSTACK action yields Figure 4. Note that the lower block is still an uninstantiated variable.

If every goal requires a node expansion, planning would never terminate. Fortunately, there is another process, called linking, that also achieves a goal. Linking is performed when the goal is "already true" somewhere in the plan, i.e., when the goal matches an assertion of another node in the plan. That other node must not be ordered below the goal, however. In this case (ON C y) will match (ON C A) in the start node, giving an instantiation for y. N8 becomes a "phantom node" and is ordered below the start node. In addition, a protection relation (PROTECTED (ON C A) START N7) is recorded. Actually, the planner always first tries to achieve a goal by linking and does a node expansion only when linking is impossible. Why should an agent have to work for the goal when it has already been achieved? In general, there may be several possible linking alternatives, just as there may be several possible expansions. After linking N8 and N9 and expanding N5 with a STACK action, the diagram is as shown in Figure 5. Implied by the linking process is the ability to retrieve literals asserted by any node in the plan that match a given pattern.

At this point in plan synthesis a conflict is detected. A conflict is the situation where two unordered nodes assert contradictory literals. Here N4 asserts (CLEAR C) and N5 asserts (NOT (CLEAR C)). (Recall that the assertions of an action node are the instantiated action postconditions.) A conflict is symptomatic of a contradictory or inconsistent world state. It is resolved by ordering the two nodes while respecting the protection relations of the upper node. If the planner does not resolve conflicts, it becomes indeterminate whether an assertion is true at a particular point in the plan, and linking is impossible. The general situation is illustrated in Figure 10. Here P and (NOT P) are the contradicting assertions, with P being protected between NA and NC and (NOT P) being protected between NB and ND. There are two possible resolutions of the conflict: order NC above NB or order ND above NA. Just ordering NA directly above NB or NB directly above NA would violate the protection relations and produce an invalid plan. If there are no protection relations on the contradicting

Protection Relations: (PROTECTED P NA NC)  
(PROTECTED (NOT P) NB ND)

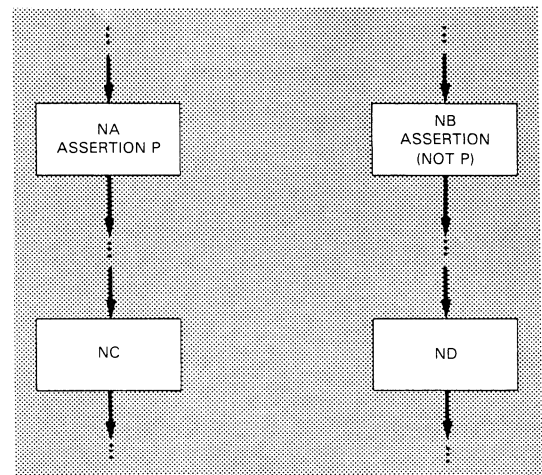


Figure 10. A conflict situation.

literals, as is the case at this point in the example, a simple ordering of the contradicting nodes is satisfactory. The planner orders N4 above N5. There is also a second conflict at this point: (CLEAR C) in N10 and (NOT (CLEAR C)) in N7. This is resolved by ordering N7 above N10. The plan diagram now looks like Figure 6. The planner looks for conflicts after every node expansion and variable instantiation. Until Figure 5 there were no conflicts.

The three fundamental operations of the planner have now been illustrated: linking, node expansion, and ordering to resolve conflicts. To reach a complete solution merely requires a repetition of these. The complete sequence of planning steps is given in Table 2. The plan is complete when the goal stack has been emptied and all conflicts are resolved. This state is reached in Figure 8. Removing the phantom-node scaffolding from this diagram yields Figure 9, the actual plan.

Although this blocks-world problem does not illustrate expansion with inference rules or linking to derived assertions, these are now briefly explained. Recall that derived assertions are now true at the moment when one of their preconditions is contradicted, whereas ordinary assertions continue to hold until they are explicitly contradicted. This means that goal-protection relations involving derived assertions must be iteratively "extended" during planning using the following protection transitivity rule:

If (PROTECTED literal1 node1 node2) and  
 (PROTECTED derived.literal node2 node3) Then  
 Record (PROTECTED literal1 node1 node3)

Whenever a new protection relation is recorded, the planner must attempt to extend it using this rule. The net result is

**Table 2. Major Planning Steps in Solving Blocks-World Problem**

1. Expand goal N4, (ONTABLE C), into PUTDOWN(C). See Figure 3.
2. Expand goal N7, (HOLDING C), into UNSTACK(C,y). See Figure 4.
3. Link goal N8, (ON C y), to the Start node by applying substitution {A/y}.
4. Link goal N9, (CLEAR C), to the Start node.
5. Expand goal N5, (ON B C), into STACK(B,C). See Figure 5.
6. Order N4 and N5 to resolve the conflict over (CLEAR C) in N4 and (NOT (CLEAR C)) in N5.
7. Order N7 and N10 to resolve the conflict over (CLEAR C) in N10 and (NOT (CLEAR C)) in N7. See Figure 6.
8. Link goal N10, (CLEAR C) to N4.
9. Expand goal N11, (HOLDING B), into PICKUP(B). See Figure 7.
10. Link goal N12, (ONTABLE B), to the Start node.
11. Link goal N13, (CLEAR B), to the Start node.
12. Expand goal N6, (ON A B), into STACK(A,B).
13. Order N5, N6 to resolve the conflict over (CLEAR B) in N5 and (NOT (CLEAR B)) in N6.
14. Order N11 and N14 to resolve the conflict over (CLEAR B) in N14 and (NOT (CLEAR B)) in N11.
15. Link goal N14, (CLEAR B), to N5.
16. Expand goal N15, (HOLDING A), into PICKUP(A).
17. Order N7 and N15 to resolve the conflict over (CLEAR A) in N7 and (NOT (CLEAR A)) in N15.
18. Link goal N16, (ONTABLE A), to the Start node.
19. Link goal N17, (CLEAR A), to the Start node. See Figure 8.

that nonderived preconditions of inference rules are protected over the region of the plan in which a derived assertion must hold. Also, inferences must be allowed to chain forward spontaneously, like event rules, so that all conflicts involving derived assertions can be detected. These conflicts are resolved like any other. Because it is relatively expensive to monitor the triggering of forward-chaining rules, use of inferences and derived assertions in planning should be minimized.

Finally, backtracking must be mentioned. If for a particular goal no linking is possible and no expansion exists, the goal cannot be satisfied. The planner has reached an impasse and must backtrack up the search tree and reconsider one of its earlier decisions. In this blocks-world example no backtracking is required, but it is routine in planning for real-world applications.

### Action Selection

It is frequently the case that more than one action is a candidate for expanding a goal node. How does the planner decide which to try first? There are two general possibilities: domain-independent criteria and domain-specific advice.

Useful domain-independent criteria for rating candidate actions are the number of nonlinkable preconditions and the number of "bonus goals." In the blocks-world planning example there were two expansions for N7: UNSTACK(C) and PICKUP(C). At step 2 both the preconditions of UNSTACK can be linked, and for PICKUP the precondition (ONTABLE C) would need to be expanded. This extra expansion would mean more work for both the planner and the agent executing the plan. Thus, UNSTACK looks like the better choice.

Every node expansion is undertaken to satisfy the current top goal in the goal stack. Sometimes the new action will, by virtue of its other consequences, serendipitously achieve additional goals in the stack. These are bonus goals, which the planner will be able to satisfy merely by linking. If two expansion candidates have the same number of unlinkable preconditions, the one that achieves more bonus goals will likely result in a simpler plan. A reasonable heuristic static-evaluation function for rating expansion candidates is

$$(F * \text{bonus-goal-count} - \text{nonlinkable-precondition-count}).$$

where F is a factor greater than 1. It is desirable that F approximate the average number of nonlinkable preconditions expected for actions in a particular domain. This domain-independent evaluation function quantifies the doctrine that the best action is one that maximally reduces the difference between the desired state, represented by the unsatisfied goals, and the present planning state, represented by the start-node assertions plus the assertions of actions already planned. It favors actions that achieve several goals at once over those that achieve just one.

Domain-dependent criteria for action selection may be implemented by adding additional preconditions to the action description. For example, if there are two competitive actions, PICKUP-BY-HAND and PICKUP-WITH-FORKLIFT, for lifting objects, it might be desirable to "advise" the planner to discriminate on the basis of weight. PICKUP-BY-HAND could be given two additional preconditions, (WEIGHT-IN-LBS object w) and (LESSP w 40). PICKUP-WITH-FORKLIFT would be given a similar condition requiring a weight of at least 40 lb. The second precondition is actually an example of an action constraint, which is discussed below. This approach has the

advantage of not requiring additional machinery in the planner beyond that needed for "ordinary" preconditions. However, for logical clarity in complex domains it may be preferable to factor out the control preconditions into distinct "metarules" for managing such control decisions (5).

An alternative to mechanical decision making in selecting an action is for the planner to ask a human operator to make the choice. This is the interactive approach to planning exemplified by SIPE (6). By keeping a human operator in the loop, the possibility of the planner getting lost in search is greatly reduced. SIPE also provides a number of other custom features such as a constraint language for constraining the value of uninstantiated variables and nonconsumable resource-management procedures that allow action resources to be specially declared in the action description rather than with precondition and postcondition literals. A recent version also has sophisticated procedures for recovering from execution errors (7).

### Constraints

As seen in the PICKUP example above, it is highly desirable to be able to place constraints on the variables of an action rule. What exactly is a constraint? For present purposes it may be defined as any EVALuable LISP form taking action variables as arguments. Normally evaluation is delayed until all variables have been instantiated. If evaluation returns a non-NIL value, the constraint is satisfied. Such a feature provides an entry into conventional algorithmic computation and also allows the planner to carry along partial descriptions of uninstantiated variables without a premature commitment (8). Constraints that are not completely instantiated in the course of plan generation may require a special constraint-satisfaction (qv) package to select values for remaining variables.

In some planning domains it may be permissible to violate selected constraints. A recent version of GARI (9) provides for both "hard" and "soft" constraints invoked by advice rules rated with a numerical priority. The rules are applied in order of decreasing priority. If an inconsistent plan results, the system backtracks to the most recent lowest-priority constraint application and discards it. The procedure is shown to satisfy an optimality property, and feasibility has been demonstrated on plans for the machining of mechanical parts.

### Inhibiting Backward Chaining

Not all preconditions of an action should be expanded to make them true. For example, in the UNSTACK action the precondition (ON upper-block lower-block) should already be true when the unstacking occurs since it is pointless to stack two blocks just so they can be unstacked. Some preconditions should be achieved only by linking. Tate called these "hold" conditions (4). Contemporary planners commonly have syntactic conventions for indicating when backchaining should be inhibited. This helps to pare down the number of candidate node expansions and reduce aimless search during backtracking. Otherwise, even if the condition can be achieved initially by linking, on backtracking over the linking operation the planner may try a node expansion when this would be pointless.

### Hierarchical Planning with Macro Actions

Programmers writing in algorithmic languages such as Ada and LISP find it convenient to group statements into subrou-

tines and functions. For similar reasons, the author of an action knowledge base for a planner may find it convenient to "program" primitive actions into macro actions. As with subroutines, the elements of a macro action may themselves be macro actions. The analogy to LISP macros is even better since macro actions are usually completely expanded prior to execution. In addition to organizational convenience, planning with macro actions may help to reduce search. In the case where each macro action has just one expansion, search is eliminated entirely, and the task of writing the action descriptions takes on the character of ordinary programming. NOAH, NONLIN, SIPE, and MOLGEN are examples of planners that can synthesize plans from macro actions.

Protection relations are necessary to tie together the structure of macro actions, and there may be precondition goals to be satisfied, just as for primitive actions. Allowing embedded goals (in addition to preconditions at the "top") gives added power. Figure 11 shows a diagram for the macro action DECORATE in the domain of home construction (4). The dotted lines signify protection relations. The top-level action in this domain is BUILD-HOUSE, which expands into 10 intermediate-level actions such as LAY-BRICKWORK, FINISH-GRADING, and DECORATE. All the components of DECORATE happen to be primitive actions. For example, the primitive action FINISH-CARPENTRY is defined:

```
(FINISH-CARPENTRY ())
() → ((CARPENTRY-FINISHED)))
```

By writing macro actions, the author of the action knowledge base is able to control the sequence of actions as well as suppress much of the state-description detail in modeling the construction world. This can be a convenience in stereotyped circumstances, but there are also disadvantages.

In a system that constructs plans from macro actions, both

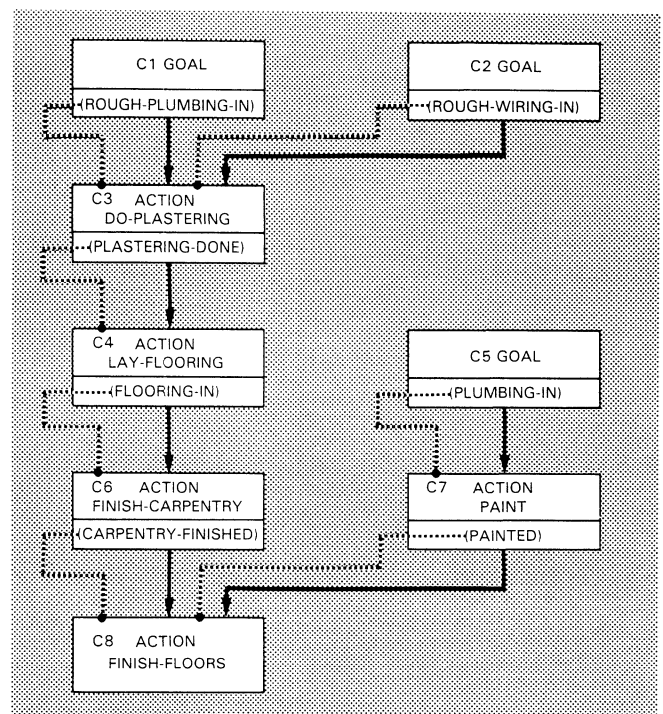


Figure 11. The DECORATE macro action.



goal nodes and action nodes may be expanded into either primitive actions or macro actions. In expansion with a macro action, successors of the expanded node generally become successors of the bottom node(s) of the macro action, in this case C8. [However, SIPE (6) provides more options here.] Predecessors of the expanded node become predecessors of the "top" nongoal nodes in the expansion, in this case C3 and C7.

This topic brings out an issue in planning philosophy. A planner knowledge base that merely specifies alternative expansions of high-level actions into lower-level actions, without reference to underlying state changes, is effectively a hierarchical set of recipes. For example, the rule base may specify an expansion for the action "reseed lawn" as the sequence: mow grass very short, rake up loose debris, distribute seed, cover with topping mulch, distribute fertilizer, roll with a lawn roller, and water frequently. Although this plan suffices for the amateur gardener, it is not based on a deep understanding of plant physiology and soil chemistry. If the knowledge-base author provides for several levels, with alternatives and meta-level control, the rote nature of this style of planning may not be apparent or even harmful. In fact, if the underlying theory of a domain is poorly understood or unnecessarily complex, as in this gardening example, empirical expansions are the method of choice. This approach can also help to avoid the AI nemesis of uncontrolled backtracking. On the other hand, plans constructed in this way may exhibit less flexibility and novelty because the planner can only assemble and play back action patterns that have been written into the knowledge base. And if during execution a sequence goes awry, empirical expansions offer little guidance in fixing the problem. Thus, the general issue of deep knowledge vs. shallow knowledge applies to planners as well as to other knowledge-based systems.

Hierarchical planning with macro actions normally proceeds in a top-down manner. Another planning control paradigm has been proposed, opportunistic planning, in which control jumps about between levels, sometimes working bottom-up and sometimes top-down as the plan is assembled (10). These ideas derived from studying verbal protocols of human subjects planning multiple-goal errand-running problems ("stop at the vet," "visit the bookstore," etc.) in a hypothetical town. Using a sophisticated blackboard control architecture (see Blackboard systems), pattern-invoked "knowledge sources" were written that successfully simulated the planning behavior of a human subject on an errand-planning problem. This may be a promising new direction for planning control if robustness and general applicability can be demonstrated on real-world problems.

### Procedural Subplanners

For certain specialized planning tasks, such as route planning (see Path planning and obstacle avoidance), it may be much more efficient to invoke a procedural subplanner (11). This is a domain-specific subsystem that aids in planning the achievement of a specific class of goals. For example, in the robot domain the goal (INROOM ROBOT *x*) would be associated with a procedural route-planning subplanner that could apply special search algorithms and geometric knowledge. When an INROOM goal is encountered that is not linkable, the subplanner is invoked to generate a primitive or macro action into which the goal is expanded. A macro action might specify a sequence of traveling actions or intermediate location goals that would achieve the original INROOM goal.

### Goal Hierarchies and Goal Ordering

Planning is inherently sensitive to the order in which goals are attempted. This applies both to the original goal set as well as to action preconditions, which become subgoals upon node expansion. An adverse ordering can dramatically increase the amount of search required to find an acceptable plan and can even prevent a solution from ever being found, regardless of search time.

To understand the problem, consider a domain with a midget household robot that must stand on a chair to reach a light switch. Assume that chairs are too heavy for the robot to move. The robot's turn-off-light action will have preconditions (ON ROBOT chair) (NEXTTO chair LIGHTSWITCH). Suppose further that there are many chairs, only one of which is near the lightswitch. With the above ordering of preconditions the planner will probably pick the wrong chair and have to backtrack many times until it links the first goal to the right chair. If the order of the goals is reversed, the planner will not have to backtrack at all. Analogous phenomena are also seen in constraint-satisfaction (qv) and database-retrieval problems. The incorrect ordering can also lead to backtracking over large segments of a plan rather than just over a linking. To see how an adverse ordering can prevent a solution entirely, suppose now that the robot can push chairs about, and in the initial state it is on a chair, say CHAIR1, not near the switch. The planner achieves the first goal by linking to the initial state and then protects (ON ROBOT CHAIR1). To achieve the second goal, the robot must be on the floor to enable it to push some chair NEXTTO the switch. However, protection of the first goal pins the robot on CHAIR1, preventing the second goal from being achieved at all, and no solution plan is found. Again, reversing the order of the goals enables a solution. (Note: some planners have a limited ability to surmount this simple difficulty by restoring a phantom back to a goal but still experience the phenomenon in more complex situations.)

There exist several remedies for this problem. One, already suggested, is for the planner to permute the order of its goals (and subgoals!) if it fails to find a solution. However, except in small, toy domains, this will lead to intolerable computation times. A much better idea is to establish a hierarchy or priority scheme on the literals of a domain (12,13). At the top level are literals that are most difficult to achieve, determine the essential structure of the domain's state space, or are not affected by any plannable action. At the bottom of the hierarchy are the easy, "detail"-level literals. The planner's goal stack is segregated according to this hierarchy, just as the job queue of an operating system is ordered by job priority. All goals at level *n* are achieved before going to work on level *n* + 1 goals. In this way the planner is able to generate a broad-brush plan at an abstract level and then progressively refine the plan as goals at the lower levels are achieved. In a mobile-robot domain Sacerdoti (13) defined the following hierarchy:

- Level 1: TYPE, COLOR
- Level 2: INROOM
- Level 3: PLUGGED-IN, UNPLUGGED
- Level 4: NEXTTO

The type and color of an object cannot be changed by any action and so are the most critical. NEXTTO is the easiest, requiring only a movement within a room.

Some permutation of goal ordering within a given level may still be required to ensure a thorough search for an acceptable plan. Thus, goal hierarchies can greatly reduce but in general cannot eliminate the need to permute goals. Although most domains are probably amenable to the hierarchic approach, the process of sorting the literals of a domain into classes is at present an art, and no algorithmic procedure is known. The penalty for a poorly chosen hierarchy may be unnecessary search or overlooked solutions.

Splicing (14) is a complete, but more radical, solution to the goal-order problem. It is a violent form of conflict resolution, in which goal-protection relations are intentionally violated, and the violated goals are "demoted." Demotion may involve anything from changing a phantom node back to a goal to the recursive erasure of large sections of the plan already generated. Splicing allows a planner to do a thorough search much faster than by permuting goals, but the performance improvement has not been determined analytically. The subplan-erasure procedures of the splicing process also apply to other situations where an existing plan must be revised. Examples are in recovery from an execution error and adaptation of an existing plan to a new, but similar, situation.

### Temporal Planning

The planners discussed above know nothing of numerical time. Actions are assumed to be instantaneous, and there is no way to impose time constraints on actions or goals. A temporal planner called DEVISER (15), an extension of NONLIN, overcomes some of these limitations. In the DEVISER system every action has a duration, which may be zero, and a start time interval called the window, which bounds the point in time when the action may begin. The default window is  $(0 \infty)$ . As the plan develops, these windows are compressed as necessary so that time constraints are observed. Other features of DEVISER include scheduled events, forward-chaining spontaneous events, and packaged goals that may have time constraints on their achievement and an achievement duration. When planning terminates, the system schedules each action by selecting a start time from within the final window.

The following FILL-CYLINDER action description shows the utility of durations:

```
(FILL-CYLINDER (capacity cylinder old-level rate)
  ((FLOW-RATE rate)
   (CAPACITY cylinder capacity)
   LEVEL cylinder old-level))
→
((LEVEL cylinder capacity)
 (NOT (LEVEL cylinder old-level)))
(DURATION ((capacity - old-level) * rate))).
```

When two actions are ordered, their windows may need to be compressed. Each compression may induce a ripple of compressions of neighboring nodes in the plan diagram, spreading out from the point of the ordering. Suppose  $(EST1 \ LST1)$  and  $(EST2 \ LST2)$  are, respectively, the windows of action nodes  $N1$  and  $N2$ , and  $D1$  is the duration of  $N1$ . If  $N1$  and  $N2$  are ordered, it may be necessary to decrease  $LST1$  and increase

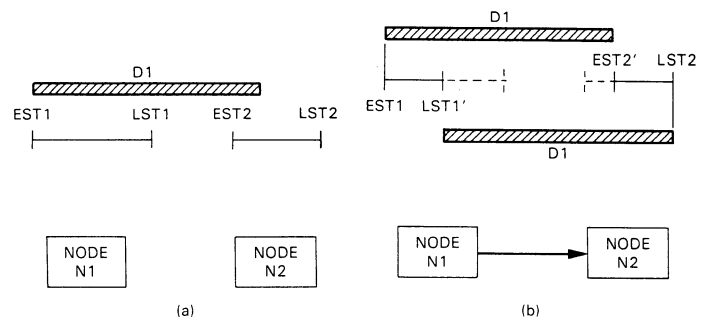


Figure 12. Window compression before and after ordering  $N1$  and  $N2$ .

$EST2$  in order to maintain the inequalities  $EST1 + D1 \leq EST2$  and  $LST1 + D1 \leq LST2$ . Figure 12 shows how the duration  $D1$  acts to compress the two windows when ordering occurs. Of course, if  $D1$  is too long,  $LST1'$  may become less than  $EST1$ , indicating a time constraint violation, and the planner must backtrack.

Consecutively is another temporal constraint even stronger than ordering. Two actions  $N1$  and  $N2$  are consecutive if  $N2$  begins immediately on termination of  $N1$ , i.e.,  $EST1 + D1 = EST2$  and  $LST1 + D1 = LST2$ . Operationally, this means that if the window of one node is adjusted by  $\Delta t$ , the window of the other must be adjusted by the same amount. Allen and Koomen (16) have formulated a complete set of seven temporal relationships that may exist between actions: before (ordered), equal, meets (consecutive), overlaps, during, starts, and finishes.

### Practical Applications of Planning

Although no AI planner is known to be in routine industrial use (as of March 1986), planners are now on the verge of commercial application. The limitations in many cases are in technology transfer, politics, and availability of trained personnel. Quite a few systems have achieved convincing demonstrations of planning in a variety of real-world domains. These include DEVISER (15) in planning spacecraft activities, GARI (17) in the machining of mechanical parts, ISIS (18) in manufacturing, KNOBS (19) in Air Force tactical missions, MOLGEN (8) in genetic experiments, NONLIN in turbine overhaul and naval resupply missions (20), and SIPE (6) in aircraft-carrier-deck operations. A very impressive domain-specific system was Wesson's air-traffic-control planner (21), which in 1977 rivaled human-controller performance.

### Advanced Topics and Research Questions in Planning

There remain many open areas in planning where present understanding is weak or nonexistent. This section presents some speculative, partially overlapping advanced topics and questions for future research in planning. Their scope ranges from the master's-thesis level upward, and some are completely unexplored.

1. What is the best way to plan with stochastic or nondeterministic actions in partially unknown environments? A closely related problem is planning with information-gathering actions. The outcome of such an action is by definition unknown until the action is executed. Is it sufficient simply

to provide the planner with the ability to generate conditional plans (22)?

2. Contemporary planners do not learn and improve with practice. People do not plan anew each morning how they are going to get to work. The problem is planned once or twice, and thereafter they merely access and execute the same stored plan. How can a planner be made to improve its performance and reduce its search time with practice in a particular domain? Triangle tables (23) were an early approach to this problem, in which a linear sequence of actions is stored in flexible form so that subsequences may be applied to some new problem. However, the problem of efficient retrieval has not been solved. In the absence of efficient retrieval and screening, it may actually be faster for a planner simply to synthesize plans from scratch rather than search through a large library of old plans every time a new goal is encountered. Progress has been made on operator- (action-) selection learning (24,25). How can previously successful plans and subplans be stored, retrieved, and screened efficiently for application in new planning problems?
3. Wesson's domain-specific planner dealt with a dynamic world of aircraft flying through three-dimensional space. How can domain-independent planners be made to deal with dynamic external worlds (26)? For example, how could a planner, given a knowledge of the laws of physics, synthesize the series of rocket-engine and thruster firings that would enable a spacecraft in Earth orbit to intercept a passing comet of known trajectory?
4. "Protracted goals" are those that must be preserved for a span of time. DEVISER could achieve positive protracted goals under certain conditions, such as when a single action would suffice for the required duration. How can protracted goals be achieved when a series of actions are required? For example, suppose the goal is to maintain electrical power to a portable computer. While at home, power is provided from a wall outlet. On a short trip to a friend's house, the internal battery suffices. On a cross-country vacation in a camper, a storage battery is required. Each action (e.g., plugging into the outlet) satisfies the goal for only part of the required period.
5. Avoidance goals (27) are negative protracted goals in the above terminology. It is insufficient merely to backtrack the planner whenever it expands with an action or triggers an event that contradicts an avoidance goal. The planner must be able to intentionally insert actions that will contradict the preconditions of all possible spontaneous-event chains leading to an avoidance goal. How can planners be extended to handle general avoidance goals?
6. Discrete-event simulation is a mature technology for modeling complex domains with remarkable precision. But simulations are passive and do not achieve or protect goals. It is possible to envision a "planulator," blending the capabilities of a planner and discrete-event simulator. How can the world-modeling power of a simulation be combined with the logical-reasoning and goal-oriented capabilities of an AI planner?
7. One of the weaknesses afflicting contemporary planners is uncontrolled chronological backtracking. The doctrine of dependency-directed backtracking (qv) (28) offers guidance, but the adaptation of this idea to planners in a computationally efficient implementation is still a research prob-

lem. In fact, an inordinate fear of backtracking probably impels much of the research on elaborate control architectures. A really efficient, intelligent backtracking method for planners would greatly reduce the cost of bad decisions and lessen the imperative for infallible first-time decision processes.

8. Program synthesis and plan synthesis show striking parallels, and over the years concepts from algorithmic programming have been a source of inspiration to planning researchers. In fact, program synthesis may be viewed as the planning of action sequences for agents called computers. Synthesizing "plans" for a robot to fetch the longest piece of straight lumber in a pile and for a microprocessor to return the largest positive integer in a list are probably formally identical problems. This is a sobering perspective because program synthesis is appreciated to be immensely difficult, and yet programming is evidently only a special case of the general planning problem! Is there a useful theoretical framework within which planning, program synthesis, and program verification can be unified? (see Automatic programming).
9. Another immensely difficult topic is the generation of plans for multiple sentient agents that may have to communicate and negotiate to achieve and preserve mutual and conflicting goals (29-32). Can contemporary AI planning technology be extended into an operational theory of "computational politics?" Will it be possible to automate the U.S. House of Representatives?

#### Historical Remarks

Planning and problem solving began as a common topic with GPS (33) (in some circles these two terms are still synonymous). In GPS, differences between the present and desired state guided the selection of operators. STRIPS (34) was a major early planner that in 1969 generated linear action sequences for SRI's robot SHAKEY. It introduced the notions of add-and-delete lists, which have evolved only slightly into the present precondition- and postcondition-style rules for describing primitive actions. STRIPS MACROPS (23) were an early form of macro action limited to a sequence of primitive actions. HACKER (35) was constructed in 1973 to explore the problem of improving performance with experience. As an incidental feature, it introduced the notion of goal protection. In 1975 NOAH (36) introduced concurrency into planning and extended the notion of macro actions. It originated the ideas of conflict detection and resolution by ordering. NONLIN (3,4) improved on the NOAH model by adding goal structure, backtracking, and linking. NOAH did not do linking but rather planned each goal back to the initial state, applying heuristic "critics" to simplify the plan. Subsequent experience suggests that the linking procedure is to be preferred since simplification critics have so far not proven robust on a wide spectrum of problems.

#### BIBLIOGRAPHY

1. C. Green, Theorem-Proving by Resolution as a Basis for Question-Answering Systems, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 4, Halsted, Wiley, New York, pp. 183-205, 1969.
2. J. McCarthy and P. J. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, in B. Meltzer and D.



- Michie (eds.), *Machine Intelligence*, Vol. 4, Halsted, Wiley, New York, pp. 463–502, 1969.
3. A. Tate, Generating Project Networks, *Proceedings of the Fifth IJCAI*, Cambridge, Mass, pp. 888–893, 1977.
  4. A. Tate, Project Planning Using a Hierarchic Non-Linear Planner, Report No. 25, Artificial Intelligence Department, University of Edinburgh, August 1976.
  5. R. Davis, "Meta-rules: Reasoning about control," *Artif. Intell.* 15(3), 179–222 (December 1980).
  6. D. E. Wilkins, "Domain-independent planning: Representation and plan generation," *Artif. Intell.* 22(3), 269–301 (April 1984).
  7. D. E. Wilkins, Recovering From Execution Errors in SIPE, Technical Note 346, AI Center, SRI International, Menlo Park, CA, January 1985.
  8. M. Stefik, "Planning with constraints (MOLGEN: Part 1)," *Artif. Intell.* 16(2), 111–139 (May 1981).
  9. Y. Descotte and J. C. Latombe, "Making compromises among antagonist constraints in a planner," *Artif. Intell.* 27(2), 183–217 (November 1985).
  10. B. Hayes-Roth et al., Modelling Planning as an Incremental, Opportunistic Process, *Proceedings of the Sixth IJCAI*, Tokyo, Japan, pp. 375–383, 1979.
  11. R. E. Fikes, Knowledge Representation in Automatic Planning Systems, in J. K. Jones (ed.), *Perspectives in Computer Science*, pp. 63–75, 1977.
  12. L. Siklossy and J. Dreussi, An Efficient Robot Planner Which Generates its own Procedures, *Proceedings of the Third IJCAI*, Stanford, CA, pp. 423–430, 1973.
  13. E. D. Sacerdoti, "Planning in a hierarchy of abstraction spaces," *Artif. Intell.* 5(2), 115–135 (1974).
  14. S. A. Vere, Splicing Plans to Achieve Misordered Goals, *Proceedings of the Ninth IJCAI*, Los Angeles, CA, pp. 1016–1021, 1985.
  15. S. A. Vere, "Planning in time: Windows and durations for activities and goals, *IEEE Trans. Patt. Anal. Mach. Intell.* PAMI-5(3), 246–267 (May 1983).
  16. J. Allen and J. Koomey, Planning Using a Temporal World Model, *Proceedings of the Eighth IJCAI*, Karlsruhe, FRG, pp. 741–747, 1983.
  17. Y. Descotte and J.-C. Latombe, GARI: A Problem Solver that Plans How to Machine Mechanical Parts, *Proceedings of the Seventh IJCAI*, Vancouver, B.C., pp. 766–772, 1981.
  18. M. S. Fox, Constraint-Directed Search: A Case Study of Job-Shop Scheduling, Ph.D. Dissertation, Computer Science Department, Carnegie-Mellon University, October 1983.
  19. C. Engleman et al., KNOBS: an Integrated AI Interactive Planning Architecture, *Proceedings of the AIAA Computers in Aerospace Conference*, Hartford, CT, pp. 450–462, October 1983.
  20. A. Tate and A. M. Whiter, Planning with Multiple Resource Constraints and an Application to a Naval Planning Problem, *Proceedings of the First Conference on AI Applications*, Denver, CO, pp. 410–416, 1984.
  21. R. B. Wesson, Planning in the World of the Air Traffic Controller, *Proceedings of the Fifth IJCAI*, Cambridge, MA, pp. 473–479, 1977.
  22. D. H. D. Warren, Generating Conditional Plans and Programs, *Proceedings of the AISB Conference*, University of Edinburgh, pp. 344–354, 1976.
  23. R. E. Fikes et al., "Learning and executing generalized robot plans," *Artif. Intell.* 3, 251–288 (1972).
  24. T. M. Mitchell et al., Learning Problem-Solving Heuristics Through Practice, *Proceedings of the Seventh IJCAI*, Vancouver, B.C., pp. 127–134, 1981.
  25. D. Kibler and B. Porter, Episodic Learning, *Proceedings of the Third AAAI Conference*, Washington, DC, pp. 191–196, 1983.
  26. G. G. Hendrix, "Modelling simultaneous actions and continuous processes," *Artif. Intell.* 4, 145–180 (1973).
  27. D. A. McDermott, "A temporal logic for reasoning about processes and plans, *Cog. Sci.* 6, 101–155 (1982).
  28. J. de Kleer et al., Explicit Control of Reasoning, in P. H. Winston and R. H. Brown (eds.), *Artificial Intelligence: An MIT Perspective*, Vol. 1, MIT Press, Cambridge, MA, pp. 93–116, 1979.
  29. M. Georgeff, A Theory of Action for MultiAgent Planning, *Proceedings of the Fourth AAAI Conference*, Austin, TX, pp. 121–125, 1984.
  30. D. E. Appelt, Planning Natural-Language Utterances, *Proceedings of the Second AAAI Conference*, Pittsburgh, PA, pp. 59–62, 1982.
  31. R. C. Schank and R. P. Abelson, *Scripts, Plans, Goals and Understanding*, Erlbaum, Hillsdale, NJ, 1977.
  32. R. Wilensky, *Planning and Understanding*, Addison-Wesley, Reading, MA, 1983.
  33. A. Newell and H. A. Simon, GPS, a Program that Simulates Human Thought, in E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, pp. 279–293, 1963.
  34. R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving," *Artif. Intell.* 2(3/4), 189–208 (Winter 1971).
  35. G. J. Sussman, *A Computer Model of Skill Acquisition*, American Elsevier, New York, 1975.
  36. E. D. Sacerdoti, *A Structure for Plans and Behavior*, Elsevier North-Holland, New York, 1977.

#### General References

- L. Daniel, Planning and Operations Research, in T. O'Shea and M. Eisenstadt (eds.), *Artificial Intelligence: Tools, Techniques, and Applications*, Harper & Row, New York, pp. 423–452, 1984.
- N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, Chapters 7 and 8, 1980.
- E. D. Sacerdoti, Problem Solving Tactics, *Proceedings of the Sixth IJCAI*, Tokyo, Japan, pp. 1077–1085, 1979.

S. VERE  
Lockheed AI Research Center

#### POLITICS

A system that simulates human ideological understanding of international political events, POLITICS was written in 1979 by Carbonell at Yale University and is a successor to MARGIE and a predecessor to BORIS (qv) (see J. Carbonell, POLITICS, in R. C. Schank and C. K. Riesbeck (eds.), *Inside Computer Understanding: Five Programs Plus Miniatures*, Lawrence Erlbaum, Hillsdale, NJ, pp. 259–307, 1981).

M. TAIE  
SUNY at Buffalo

#### POP-2

POP-2 is a programming language developed at the University of Edinburgh and has achieved wide use in Europe. It shares certain similarities to the language LISP (qv) in that it uses lists as important data structures, yet differs to the extent that programs and data do not share the same syntax. POP-2 adopted a more ALGOL-like syntax. More recently, POP-2 has been extended into the POP-11 and POPLOG programming systems (see R. Burstall, D. Collins, and R. J. Popplestone,

*Programming in POP-2*, Edinburgh University Press, 1971, and R. J. Popplestone, *The Design Philosophy of POP-2*, in D. Michie (ed.), *Machine Intelligence*, Vol. 3, American Elsevier, New York, pp. 393–402 (1968). For an overview of POPLOG and POP-11 see S. Hardy, *A New Software Environment for List-Processing and Logic Programming*, in T. O'Shea and M. Eisenstadt (eds.), *Artificial Intelligence: Tools, Techniques, and Applications*, Harper & Row, pp. 110–136, 1984).

J. ROSENBERG  
SUNY at Buffalo

## PRIMITIVES

Primitives are philosophical atoms, usually corresponding to words in a natural language, such as English, and are said to name basic concepts underlying human thought. They are made use of in knowledge-representation (qv) programs and, more specifically, in programs for representing and analyzing the meaning of natural-language expressions. The notion of primitive is historically close to that of atom: both are entities of which other, more complex, entities are formed (in this case, definitions or symbolic knowledge representations) and that cannot themselves be broken down or defined.

In AI, primitive-based systems made their first appearance in the sixties, with the systems of Wilks (1), Schank's system for verb decomposition (2), Norman and Rumelhart (3), and Joshi (4), but the advocacy of such entities as essential to a computational analysis of natural-language structure can be traced back at least to Masterman (5).

A simple example of a primitive would be Schank's TRANS (later subdivided into MTRANS, PTRANS, ATRANS, and now, ATRANS1 and ATRANS2), which he considered to be a name for the single concept underlying all such acts as buying and selling on the ground that they involved the physical TRANSfer of some entity in space, later modified to a more abstract notion of legal transfer, ATRANS. The justification for seeking such a commonality between apparently different actions or entities was the same reason that had been given in linguistics: that such a generalization would aid perspicuity in processes and substitute a single rule (using the primitive) for many, based on the corresponding surface words. When discussing primitives, it is important to bear in mind that those who advocate their use were not, in general, discussing the efficacy of particular atoms in the vocabulary of a language of representation but of the complex formal structures of which the primitives were part: e.g., conceptualizations (see Conceptual dependency), and scripts (qv) in Schank's work, and formulas, templates, and pseudotexts in Wilks's system.

TRANS is untypical of primitives in that it does not look immediately like an English word (though in fact it is short for TRANSfer). It has been argued that since primitives appear to be English words (usually in uppercase), then that is what they are Zwicky (6) has argued that there are no primitives that do not have obvious "translations" into English or another language and, in that sense, there cannot be "secret, or incomprehensible, primes."

However, that was strongly denied by Katz (7), Postal (8) and Schank (9). As Katz puts it, "although the semantic markers are given in the orthography of a natural language, they cannot be identified with the words or expressions of the language used to provide them with suggestive labels." In other

words, Katz considers that primes really exist quite independently of the words used to describe, locate, or interpret them.

## Linguistic Background

The issues in AI arose from the introduction of a semantic component (see Semantics) into a Chomskyan generative grammar (see Grammar, transformational) by Katz and Fodor (10). This required what they called "semantic markers": entities like HUMAN, which would be attached in a lexicon to the sense of "bachelor," meaning an unmarried man, but not to the sense meaning a young seal at a certain phase of its development nor to the one meaning the holder of a first degree from a university. Later, within a group known as "generative semanticists," entities called "underlying verbs" were postulated as part of the meaning of words. Thus, STRIKE would be part of the underlying structure for "remind" as it appears in "Max reminds me of Pete" (8). Finally, and simultaneously with the last tendency, Fillmore (11) suggested that verbs should be derived from structures that incorporated cases, like AGENT, whose names also were used very much like those of primitives.

## Requirements for a Primitive Set

In a primitive set certain very general restrictions suggest themselves on the membership of a natural set of primitives or markers:

1. *Finitude*: A primitive set should not, e.g., contain the names of the natural numbers. More seriously, the set should be considerably smaller than the set of words whose meaning it is to encode.
2. *Comprehensiveness*: The set should be adequate to express and distinguish the senses of the word-set meanings it is to encode.
3. *Independence*: There should not be markers  $X, Y, Z$  in the set such that there is some function  $F$  such that  $X = F(Y, Z)$ . However, this will not be so easy to achieve if the members are hierarchically organized: If, e.g., the set contains ANIMATE and HUMAN, it would then be nonindependent if there were any marker like RATIONAL in it, of which one might hold that

$$\text{HUMAN} = \text{ANIMATE} + \text{RATIONAL}$$

4. *Noncircularity*: There should not be such nonindependencies such that two markers or primitives can be mutually defined, as in  $X = F(Z, Y)$  and  $Z = F(X, Y)$ .
5. *Primitiveness*: No marker subset should be such that it could plausibly be replaced by a smaller defining set as in  $A = F_1(P, Q), B = F_2(P, Q), C = F_3(P, Q)$ .

But these desiderata for a set of primitives bring one no closer to a satisfactory definition, even a provisional one, of what a primitive is. Here is a provisional definition: "A PRIMITIVE (or, e.g., rather a set of primitives plus a syntax) is a reduction device which yields a semantic representation for a natural language via a translation algorithm and which is not plausibly explicated in terms of, or reducible to, other entities of the same type (e.g., structures of symbolic atoms)."

This definition leaves open, as it is intended to, the serious question of whether or not primitives are explicable in terms

of, or reducible to, entities of some quite other type. This is a serious question because most attacks on the use of primitives (e.g., Refs. 12–16) take the form of demands that they be explicated in terms of some other type of entity altogether; just as most bad defenses of primitives take the form of offering some very weak equivalence between primitives and other types of entities (e.g., referential objects, brain parts, abstract denotata).

Note the ubiquity of primitives in areas related to AI; the propositional calculus (see Logic, propositional) is a paradigm case with its connectives AND, OR, IMPLIES, and NOT, which constitute a primitive set on the above definition or, more correctly, any two of them constitute a primitive set that can define the other two. Although they are reducible to the single connective “|,” the Sheffer stroke, that requires considerable effort and inconvenience. It was only later, when the truth tables were derived (by Wittgenstein and Pierce), that the possibility of independent explication of the primitives arose. Only then could one give some answer to “What does AND in the propositional calculus actually mean?” Thus, if one accepts the truth tables as “giving the meaning” of the connectives, then they are no longer primitive in the sense of the above definition.

Within AI, the variable SHOPPER in Charniak’s super-market frames (16) (see Frame theory) and the Basic unit PERSON in Bobrow and Winograd’s KRL-O language (12) are both primitives. They can be bound to particular shoppers or persons, but nowhere in those systems is one told what a shopper or a person is. Nonetheless, those authors would strongly deny that there are primitives in their systems, which again shows the difficulty of agreeing on a definition of what constitutes the use of a primitive in an AI system.

### Arguments against Primitives

Hayes (13) produced a number of arguments against primitives, such as his demand to know what STUFF (the name for substance in Wilks’s system) actually meant. This is, in effect, the demand for reduction to another type of entity, in this case a model-theoretic semantics (MTS) of a primitive system. For Hayes that is closely related to the demand for MTS of programming languages, but an important difference between the two areas (programming languages and the real world) is that there is no reason to suppose that the sets and objects yielded by the semantics of a program will be the same as the sets and objects in the real world that would be the direct denotations of primitives—determined by an MTS for the area of knowledge in question, e.g., people or aircraft—used in a program that modeled some aspect of the real world. Hayes seems to assume that the two formalizations would yield the same result, but they need not since the requirements for being an adequate set of objects for a program semantics are only that they support a formal I/O equivalence for the program, and many sets might do that. But if the two sets of denotations are not the same, there will be problems for anyone who, like him, believes that such denotations are the “real meanings” of the primitives. Which set would then be the real meanings?

The demand for an MTS of a primitive system can be put in one of three analogous ways: as being like an MTS for a logic, for a programming language, or like the axiomatization of a scientific theory. In all three cases there is clear historical

evidence that the basic systems (i.e., logic, programming languages, or scientific theories) were pragmatically useful before there were axiomatizations of them. So a demand that some MTS of primitives be given before such devices can have any useful role in AI programs is inappropriate; the usefulness of logic over more than two millennia could not in any way have depended on there existing, or not existing, a formal semantics for it.

A more serious misunderstanding in Ref. 12 occurs when the authors contrast their proposal for “perspectives” in a representation (a number of points of view of the same object or action) with primitive-based approaches: “In general, we believe that the description of a complex object cannot be broken down into a single set of primitives, but must be expressed through multiple views” (12).

Primitives constitute no more than a language for the description of meaning, and therefore, a language in which alternative descriptions of the same thing can be given. Indeed, it is no accident at all that those who have been most concerned with primitives have also been concerned to emphasize the ambiguity of words, surely the paradigm of multiple description.

Bobrow and Winograd were assuming that some of the less defensible claims about primitives, such as their being the absolute and right representation of a word’s meaning, must imply that there is also as a matter of fact only one such description. But not even those who have held the strongest views about primitives have claimed this. In Schank (17), e.g., he argues that “restaurant” must have not only a narrative “scriptlike” representation in primitives but also another representation to express its “physical sense” as well. Charniak (15) is concerned with the use of the concept of case in AI and presents arguments against the use of case primitives. It should be added that there is now a body of work within AI (18,19) and linguistics (20) that claims that an MTS for the primitives of a system for analyzing natural languages can be given.

### A General Defense of Primitives

The notion of primitives in AI natural-language systems might be that they constitute not some special language, or another realm of objects, but are no more than a specialized sublanguage, consisting of words of some larger standard language, which plays a special organizing role in a language system. On that view, no claims involving notions like “ontology,” “platonism,” or “reification” are involved, i.e., the basic vocabulary of any expert system (qv) is its own domain-dependent set of primitives. Given the bias of the membership of Schank’s original set of primitives (2) toward members like EXPEL and INGEST, it could be viewed, in retrospect, not as a universal primitive system but perhaps as a domain-dependent one for bodily functions.

If one takes that view, a primitive language has no correct vocabulary, any more than English has; there may be many adequate alternative vocabularies for it, just as there may be many vocabularies for English. As Goodman (21) puts it: “In general, the terms adopted as primitives of a given system are readily definable in some other system. There is no absolute primitive, and no one correct selection of primitives.”

The clearest consequence of the thesis, that the primitive

language is also a natural language, is that there can be no direct justifications of individual primitives any more than there can be direct justifications, or nonlinguistic meanings, given for the words of English. There is no nonlinguistic realm into which one can escape from language and the explanations of English words. On this view, too, it will be hard to justify the existence of primitives that cross linguistic and cultural boundaries, even though the existence of such "universal primitives of the language of thought" has been a feature of them that has attracted many of their uses and advocates. None of this excludes the association of inferential procedures with particular primitives as partial explanations of them; just as "cause" in English may be explained to someone by producing examples of causal inference (see Inference; Reasoning, causal).

It follows from this view, in addition, that one will understand the function of primitive systems by considering the structures of real dictionaries. Statistical analyses of large dictionaries show that their constituent definitions are, in fact, statistically biased toward a restricted subvocabulary, one that is close to a natural set of primitives: cause, human, object, move, substance, etc.

Those words also appear in the dictionary itself, of course, but their definitions are wholly unsatisfactory in that it would be useless to look such words up if one did not already know what they meant in the language as a whole. Although they appear in the dictionary, they are, in a sense, without definition; their organizing role in the whole language is what matters. Sparck Jones (22) has recently published claims that go further and present computational classification procedures that range over a large dictionary and produce a candidate set of primitives, not by simply taking the most frequent words in definitions but by considering abstractly what features in the dictionary would enable word senses to be clustered into synonymous groups. Again, the standard primitives emerge as the best candidates for such features.

A fuller discussion of arguments for and against primitive elements in AI systems, particularly those concerned with natural-language comprehension, can be found in Ref. 23, along with a sample primitive vocabulary and its syntax. It should be noted, in conclusion, that there has been a movement away from emphasis on special primitives in systems and a move toward notations in which the distinction between primitives and nonprimitives (i.e., defined formal items) is blurred. This was explicitly advocated in Ref. 24 and has been extended in Ref. 25. Schank and his colleagues have, since about 1977, employed the notational device of the dollar sign (e.g., \$KID-NAP or \$PTRANS) to indicate the structure of an entity independent of its status as a primitive or defined item. Hence the issue about primitives in AI may now be a dead one, or at least dormant.

## BIBLIOGRAPHY

1. Y. Wilks, Computable Semantic Derivations, Memo 3017, Systems Development, Santa Monica, CA 1968.
2. R. Schank, "Conceptual dependency," *Cog. Psychol.* **12**, 552-631 (1972).
3. D. Norman, and D. Rumelhart (eds.), *Exploration in Cognition*, Freeman, San Francisco, CA, 1975.
4. A. Joshi, Factorization of Verbs, in C. Heidrich (ed.), *Semantics and Communication*, North-Holland, Amsterdam, pp. 271-289, 1974.
5. M. Masterman, Semantic Message Detection for Machine Translation Using an Interlingua, *Proceedings of the First International Conference on Machine Translation and Applied Language Analysis*, National Physical Laboratory, London, 1961.
6. A. Zwicky, Linguistics as Chemistry: The Substance Theory of Semantic Primes, in J. Anderson and P. Kiparsky (eds.), *Festschrift for Morris Halle*, Holt, Rinehart, and Winston, New York, 1973.
7. J. Katz, *The Philosophy of Language*, Harper & Row, New York, p. 156, 1966.
8. P. Postal, "On the surface verb 'remind'," *Ling. Inq.* **1**, 37-120 (1970).
9. R. Schank (ed.), *Conceptual Information Processing*, North-Holland, Amsterdam, p. 8, 1975.
10. J. Katz and J. Fodor, "The structure of a semantic theory," *Language* **39**, 170-210 (1963).
11. C. Fillmore, The Case for Case, in E. Bach and R. Harms (eds.), *Universals in Linguistic Theory*, Holt, Rinehart and Winston, New York, pp. 1-88, 1968.
12. D. Bobrow and T. Winograd, "An overview of KRL, a knowledge representation language," *Cog. Sci.* **1**, 3-46 (1977).
13. P. Hayes, Some Issues and Non-Issues in Representation Theory, *Proceedings of the AISB Conference*, University of Sussex, Sussex, pp. 119-125, 1974.
14. T. Winograd, On Primitives, Prototypes, and Other Semantic Anomalies, *Proceedings of the Second Conference on Theoretical Issues in Natural Language Processing*, University of Illinois at Champaign-Urbana, pp. 46-48, 1978.
15. E. Charniak, A Brief on Case, Memo. 22, Institute for Semantic and Cognitive Studies, Castagnola, Switzerland, 1975.
16. E. Charniak, Organization and Inference, *Proceedings of the Conference on Theoretical Issues in Natural Language Processing*, BBN, Cambridge, MA, 1975.
17. Reference 9, p. 26
18. K. Konolige, A Deductive Model of Belief, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 377-381, 1983.
19. R. Moore, Reasoning About Knowledge and Action, Report No. 191, SRI Artificial Intelligence Center, Stanford, CA, 1980.
20. C. Heidrich, Should Generative Semantics be Related to Intensional Logic? in E. Keenan (ed.), *The Formal Semantics of Natural Language*, Cambridge University Press, Cambridge, UK, pp. 92-111, 1975.
21. N. Goodman, *The Structure of Appearance*, Harvard University Press, Cambridge, MA, 1951.
22. K. Sparck Jones, *Synonymy and Semantic Classification*, Edinburgh University Press, Edinburgh, 1986.
23. Y. Wilks, "Good and bad arguments about semantic primitives," *Commun. Cog.* **8**, 201-219 (1977).
24. Y. Wilks, "Making preferences more active," *Artif. Intell.* 1978.
25. R. Schank and R. Abelson, *Scripts, Plans and Goals*, Erlbaum, Hillsdale, NJ, 1977.

Y. WILKS  
New Mexico State University

**PROBABILISTIC PROGRAMS.** See Autonomous vehicles; Medical-advice systems; Reasoning, plausible.

## PROBLEM REDUCTION

Problem reduction is a method of decomposing a problem into subproblems that have known solutions. In this approach to problem solving an initial problem description, a set of operators, and a set of primitive problems are given.

The problem-solving process proceeds from the initial problem until it is decomposed into primitive problems. A primitive problem is a problem whose solution is known. An operator transforms a problem into a set of subproblems. For any given problem there may be many operators that are applicable. Each of them produces an alternative decomposition of the problem into subproblems; to produce a set of problems that are all solvable, several operators might have to be tried. Solving a problem can be equated with finding an appropriate finite set of applicable operators that decompose the problem into primitive problems.

For example, the problem of going from Minneapolis to New York can be represented as a combination of getting to the airport, taking a flight, and getting to downtown New York from the airport. Each of those three subproblems can be decomposed into other subproblems. For instance, to get to the airport, one could either drive or take a cab. To get to downtown New York, one could take a cab, take the subway, or get a ride from a friend living there. The original problem is decomposed into a conjunction of subproblems (all of which have to be solved to solve the problem) and alternative subproblems (only one of them has to be solved to solve the problem).

Problem reduction could be considered a special case of problem decomposition. However, the term problem reduction in AI has always been used to indicate a general method of representing and solving problems that are described by an AND/OR graph (qv). This is the interpretation given in this entry. Indications of other types of interpretations are given below.

### Representation

Traditionally in AI a problem-reduction representation is based on the use of an AND/OR graph. An AND/OR graph provides a convenient means for keeping track of which subproblems have been attempted and which combinations of subproblems are sufficient to solve the original problem.

Suppose a problem  $P$  is represented graphically by a node. This problem may be transformed into one or more subproblems  $P_i$  that may be related to the original problem in many ways.

A relationship is indicated by a directed arc connecting two nodes. If an arc is directed from node  $n_i$  to node  $n_j$ , the node  $n_j$  is called a child of  $n_i$ , and the node  $n_i$  is called a parent of  $n_j$ . A node  $n_k$  is an ancestor of node  $n_i$  if  $n_k$  is a parent of  $n_i$  or is an ancestor of a parent of  $n_i$ . A node  $n_k$  is a descendant of node  $n_i$  if  $n_i$  is an ancestor of  $n_k$  or is an ancestor of a parent of  $n_k$ .

A node having no children is called a tip node. It can be either a node that corresponds to a primitive problem or a node to which no operators can be applied. A node representing a primitive problem is called a primitive node and a node representing a nondecomposable problem a dead-end node.

The start node of the AND/OR graph corresponds to the initial problem.

A tree is a special case of a graph in which each node has only one parent. There is a node having no parent that is called the root of the tree. It corresponds to the initial problem.

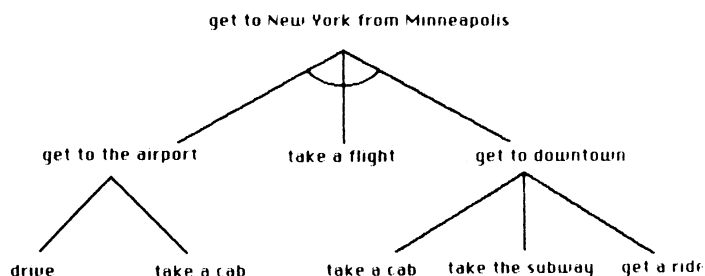


Figure 1. A travel problem.

Two common relationships between a problem and its subproblems are AND and OR.

An AND relationship between a problem and at least two subproblems exists when the solution of all of the subproblems causes the solution of the problem. An AND relationship denotes that all of the subproblems have to be solved in order for the problem to be solved. Graphically an AND relationship is indicated by a circular mark connecting all of the arcs that are part of the AND relationship.

An OR relationship between a problem and its subproblems exists when the solution of any one of its subproblems causes the solution of the problem. OR relationships denote that the solution of one of the subproblems is sufficient to solve the problem.

In the special case in which there are no AND relationships, the AND/OR graph is an ordinary graph.

The problem of getting to New York from Minneapolis described above can be represented by the AND/OR tree illustrated in Figure 1.

Each arc in an AND relationship is called an AND arc. Analogously each arc in an OR relationship is called an OR arc. In general, an AND arc and an OR arc may simultaneously point to the same node in an AND/OR graph, such as the node "take an umbrella" in Figure 2.

Many authors prefer to define AND nodes and OR nodes instead of defining AND and OR arcs. Different definitions are used in the literature and each of them presents some difficulties.

Usually a node of an AND/OR graph is defined as either an AND node or an OR node depending on the relationship that it has with its parent node (1,2). An OR node represents an alternative subproblem; an AND node represents a problem that is a member of a set of problems.

This definition of AND and OR nodes is appropriate when the problem is represented by an AND/OR tree. In the general case of an AND/OR graph things are more complex because a node can be an AND node with respect to one of its parents and an OR node with respect to another parent. For example, the node (take an umbrella) of the graph illustrated in Figure 2 is an OR node with respect to the problem (not to get wet) and an

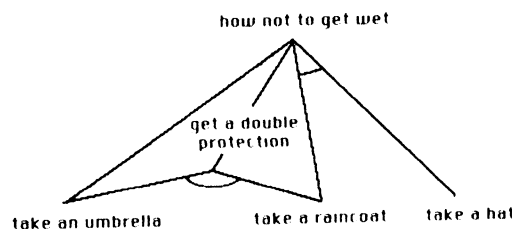


Figure 2. How not to get wet.

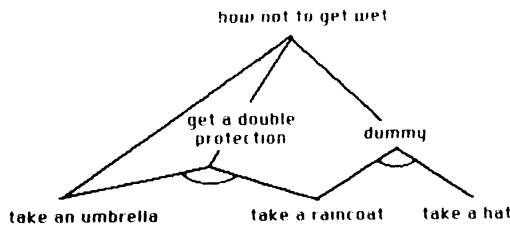


Figure 3. The AND/OR graph of Figure 2 with a dummy node.

AND node with respect to the subproblem (get a double protection).

When each node has either OR or AND relationships with its subproblems some authors (3,4) define AND nodes and OR nodes depending on the type of links that they issue. In the case in which a node has both AND and OR relationships with its subproblems, some dummy nodes are added to the graph to maintain the purity of node types. In the example shown in Figure 2 a dummy node has to be added to ensure that the node "how not to get wet" is an OR node, as illustrated in Figure 3. Dummy nodes create problems when searching for an optimal solution in AND/OR graphs because they increase the number of links to be traversed to obtain a solution. This point is covered below.

Note the difference with respect to the previous definition. A node was defined above as being an AND node or an OR node depending on the type of relationship it had with its parent, whereas here the type of the node is described as depending on the type of relationship it has with its children.

Since, in general, AND/OR graphs are more interesting than AND/OR trees, a different definition is more appropriate even though less intuitive.

An AND/OR graph could be defined as a hypergraph. Instead of arcs it has hyperarcs connecting a parent node with a set of child nodes (5,6). The hyperarcs are called connectors. Formally, an AND/OR graph  $G$  is a pair  $(N, C)$ , where  $N$  is a set of nodes and  $C$  a set of connectors  $C \subseteq \cup_{k=0}^m N^{k+1}$ , where  $N^{k+1}$  is a Cartesian product. Each  $k$ -connector is an ordered  $(k+1)$ -tuple, where the first element of the tuple is the parent and the other elements are the children. The parent node of a connector is called the input node of the connector; the children are called the output nodes. A 1-connector corresponds to an OR arc. A  $k$ -connector, with  $k > 1$  corresponds to  $k$  AND arcs from one node.

An example of an AND/OR graph is shown in Figure 4. The connector  $(n_0, n_1)$  is a 1-connector, the connector  $(n_0, n_3, n_4)$  is a 2-connector.

An important assumption made when using an AND/OR graph is that the problems are independent, and the order in

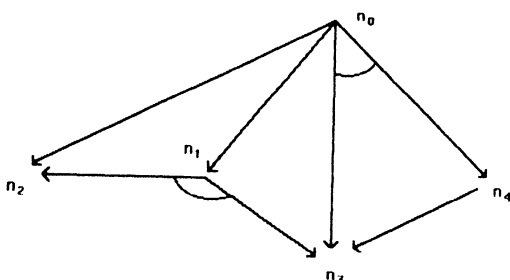


Figure 4. An AND/OR graph.

which they are solved does not matter. This is a strong assumption that limits the applicability of this problem-solving method to decomposable problems (6).

AND/OR graphs have been shown to be equivalent to context-free grammars (7). Methods to transform state-space representation into problem reduction and vice versa have been described (8). Unfortunately, transforming one representation into another does not help in finding a better representation for the given problem. This is largely an unexplored area (9,10).

### Problems Representable Using Problem Reduction

The problem-reduction approach is especially suited to problems for which the solution can be conveniently represented in the form of a tree or a graph. Strategy-seeking problems (4) are well represented with AND/OR graphs, where AND arcs represent changes in the problem caused by external conditions and OR arcs represent alternative ways of reacting to such changes.

Problems in which the solution is a partially ordered sequence of actions are good candidates for problem reduction. Typical examples are symbolic-integration problems and chemical-structures generation.

Logical reasoning and theorem proving are also good candidates for problem-reduction representation because in general the order in which subproblems are solved is not relevant.

In general, decomposable problems can be solved by the divide-and-conquer technique of problem reduction. Nondecomposable problems are more difficult because their subproblems interact.

There are cases when, even though subproblems are not independent, the set of subgoals has a unique linear hierarchy so that once one set of subgoals is solved, another set can be solved without undoing the former. An example is the Tower-of-Hanoi problem (4).

When problems share variables, a systematic way of finding all of the appropriate bindings for them should be used. Many theorem provers use the generate-and-test method to prove conjunctive goals that share variables. The difficulty with the generate-and-test approach is that an arbitrary choice of the subgoal to solve next may result in inefficiency or may even make the problem impossible to solve.

The problem of finding an appropriate ordering of the subgoals is a major problem that many authors have studied. Even though this problem has been studied for a long time, it is far from being solved. An appropriate ordering of the subgoals could reduce the search for the solution in a substantial way. Kowalsky introduced (11) the cheapest-first heuristic in which the subgoal that is cheapest to achieve is tried first. Pereira (12) showed how to take advantage of independence of subproblems to do selective backtracking.

Smith and Genesereth (13) introduced the notion of cost of a given sequence of conjuncts, and they showed how to use cost information to order conjuncts effectively. Similar techniques have been in use in the database community to optimize conjunctive queries to databases.

Nilsson (6) classified solutions to problems according to the production system used to describe them. He defined decomposable production systems to describe problems that can be represented using problem reduction.

An interesting interpretation relates Horn clauses to problem reduction (11). Horn clauses are the basis of logic pro-



gramming (qv), as done, for instance, in PROLOG. A Horn clause can be seen as a decomposition of a problem into a conjunction or a disjunction of subproblems. The procedural interpretation of Horn clauses done in PROLOG is a top-down interpretation in which a problem is decomposed into subproblems until primitive problems (called assertions) are found. Note that in this interpretation each clause is seen as a procedure in which the name of the procedure identifies the form of the problems that the procedure can solve. The body of the procedure is a set of procedure calls. Note the use of the word *set* instead of *sequence*. Procedure calls can be executed in any order or in parallel. A procedure call corresponds to invoking one of the procedures with the desired name.

The solution to the problem is obtained by doing depth-first search (qv) with backtracking (qv). Unification is used to unify terms with variables. This allows subproblems to share variables. When conjunctive goals share variables, the binding for variables found by the unification algorithm is used as long as it allows solution of successive problems. When the solution cannot proceed any further, the system backtracks and tries a different unification.

When a problem cannot be decomposed into independent subproblems, the representation with AND/OR graphs is not appropriate. Often nondecomposable problems are nearly decomposable (14). In this last case it is possible to work on each subproblem separately and to handle potential interactions between subproblems with appropriate techniques. Waldinger (15) describes a technique for achieving several goals simultaneously in which a plan is developed to achieve one of the goals and then the plan is modified to achieve the second as well. This approach is called planning (qv) in AI.

### Search

A solution to a problem represented by an AND/OR graph can be found by searching the graph. To simplify the search process, most of the algorithms make the assumption that the AND/OR graph is acyclic. This means that no node is its own ancestor. This assumption is not too restrictive because a cycle represents a circular reasoning chain that is undesirable.

There is a distinction between the graph to be searched and the graph or tree that is constructed as the search proceeds. The graph to be searched is ordinarily not explicit. It is called the search space. It can be very large or even infinite. The graph that is constructed as the search proceeds is called the search graph. It is explicit. A node is included only if a path has been discovered from the initial node to it. The search graph grows as the search proceeds.

Before describing the search process with AND/OR graphs, some additional definitions are useful.

A sequence of nodes,  $n_i, n_{i+1}, \dots, n_{i+k}$  with each  $n_j$  a child of  $n_{j-1}$  is called a path of length  $k$  from node  $n_i$  to  $n_{i+k}$ .

A node is solved if one of the following conditions holds:

- it is a primitive node or
- it is a nontip node and at least one of its outgoing connectors is connected to nodes that are all solved.

A node is unsolvable if one of the following conditions holds:

- it is a dead-end node or
- it is a nontip node and all its outgoing connectors are connected to at least one unsolvable node.

A solution graph is a hyperpath between the start node and the set of primitive nodes. It could be defined recursively:

If the start node is a primitive node, the solution graph consists only of that single node.

Otherwise, there should be one outgoing connector from the start node that connects the start node to other nodes, each of which has a solution graph. The solution graph consists of the start node, the outgoing connector, and the solution graphs for all of the nodes connected by that outgoing connector.

Informally, a solution graph of an AND/OR graph is analogous to the path of an ordinary graph. It can be obtained by starting with the starting node and by selecting exactly one outgoing connector for each node. Because a solution graph is a subgraph containing only solved nodes, the start node is solved.

In Figure 5 two solution graphs of the graph of Figure 3 are shown.

It is important to compute the cost of a solution graph. Nilsson (1) proposes two different ways of computing the cost of a solution graph. The first is called sum cost because the cost is the sum of all of the arc costs in the solution graph. The second is called max cost because the cost is the cost of the path in the solution graph having maximum path cost. The most frequently used method is the sum cost because additive AND/OR graphs have more detailed properties (3). That is the method used in the following.

In computing the cost of a solution graph, only one outgoing connector from each node is considered. Let  $n$  be the starting node. The cost associated with the node  $n$  is:

If  $n$  is a primitive node,  $c(n) = 0$ .

Otherwise, the cost of  $n$  is obtained by adding the cost of the connector  $c_n$  to the sum of the costs of the solution graphs of all of the nodes  $n_i$  connected by that connector. Assuming that the connector is a  $k$ -connector,  $c(n) = c_n + \sum_{i=1}^k c(n_i)$ .

A solution graph having minimum cost is called an optimal solution graph. The cost of an optimal solution graph is defined recursively in terms of the minimum cost of the solution graph rooted at all of the children of the node  $n$ .

Let the cost of an optimal solution graph rooted at  $n$  be denoted by  $h(n)$ :

If  $n$  is a primitive node,  $h(n) = 0$ .

If  $n$  is a dead-end node,  $h(n) = \infty$  (or a very large number).

If  $n$  is a nontip node having  $j$  outgoing connectors,  $h(n)$  is the minimum over the  $j$  connectors of the costs  $c(n)$  computed for each of the  $j$ -connectors.

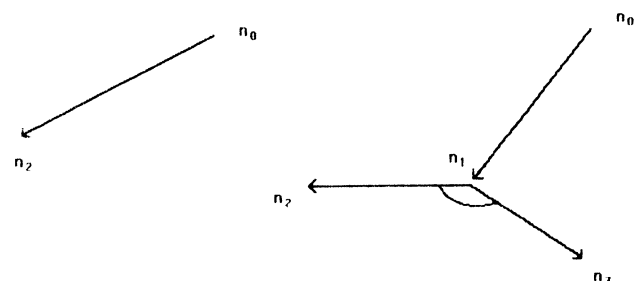


Figure 5. Two solution graphs of the graph shown in Figure 3.



**Search Algorithms.** A search algorithm can be classified as uninformed search or heuristic search (see below). The basic operation in all search algorithms is to construct a search graph by expanding nodes.

Generating all of the children of a node through all of its outgoing connectors is called expanding the node. When a child node is generated that is the same as a node generated before, an arc to the original node is created instead of creating a new node. A tip node is never expanded. A node that is not a tip node and that is not yet expanded is called an unexpanded node.

Most of the algorithms differ only in the way in which the nodes to be expanded are selected.

An algorithm that is guaranteed to find a solution graph with minimum cost if one exists is called admissible.

**Uninformed Search.** Algorithms for breadth-first and depth-first search (qv) in AND/OR graphs are similar to the breadth-first and depth-first search algorithms for state-space search. The main difference lies in the determination of the termination conditions. Every time a new node or a group of nodes is generated, the algorithm has to check whether the start node can be marked solved or unsolvable.

Nilsson (1) described the algorithms for breadth-first and depth-first search for AND/OR trees and provided suggestions for their extensions to AND/OR graphs.

**Heuristic Search.** Heuristic search can be performed with different objectives. Typically, an optimal solution (a solution with minimum cost) is desirable. Sometimes a solution that can be found with the minimum search effort (the solution that reduces the number of nodes explored) is more desirable. Often the interest is in minimizing some combination of the cost of the path and the cost of the search required to obtain the path or in minimizing the cost averaged over all of the problems.

Some task-dependent information can be used to reduce the search effort. This type of information is usually called heuristic information (see Heuristics).

During the process of search in the AND/OR graph there are nodes whose children have not yet been generated by the search process. The cost of such nodes can be estimated and the estimates used to direct the search process. Assume that for each node  $n$  there is an estimate  $h^*(n)$  of the cost of an optimal solution graph starting at  $n$ . The estimated cost is computed in much the same way as the cost:

If  $n$  is a primitive node,  $h^*(n) = 0$ .

If  $n$  is a dead-end node,  $h^*(n) = \infty$ .

If  $n$  is a nontip node whose children have not yet been generated,  $h^*(n)$  is the value of the heuristic estimate.

If  $n$  is a nontip node having  $j$  outgoing connectors,  $h^*(n)$  is the minimum of the estimated costs of all of the potential solution graphs starting at  $n$  and going through each of the  $j$ -connectors. The estimated cost of each potential solution graph is obtained by adding the cost of the connector to the sum of the estimated costs of all the nodes connected by that connector.

The heuristic search algorithm is a top-down algorithm (see Processing, bottom-up and top-down) that starts at the start node and, at each step, selects one node and expands it by generating all of its children. The algorithm associates a cost with each node.

### Top-Down Algorithm for Heuristic Search

1. Create an initial graph  $G$  consisting of only the root  $n$ . Associate with  $n$  its cost  $h^*(n)$ .
2. Repeat until  $n$  is solved or until its cost becomes  $\infty$ .
  - 2.1. Select any tip node  $n_i$  of the possible partial solution graph that is obtained by tracing down the marked connectors from  $n$ .
  - 2.2. Expand  $n_i$  by generating all of its children. If there are none, the cost of  $n_i$  becomes  $\infty$ . This means that  $n_i$  is a dead-end node. Associate with each newly generated node its cost. Label solved the nodes that are primitive nodes.
  - 2.3. Create a set of nodes  $S$  containing only  $n_i$ .
  - 2.4. Repeat until  $S$  is empty:
    - 2.4.1. Remove from  $S$  a node  $n_j$  that has no descendant in  $S$ .
    - 2.4.2. Update the cost of  $n_j$  as follows: compute the estimated cost of a solution graph starting at  $n_j$  through each outgoing connector. Set the cost of  $n_j$  to be the minimum of those costs over all of the outgoing connectors. Mark a connector through which the minimum is achieved, erasing the previous marking if different. If all of the output nodes of the marked connector are labeled solved, label  $n_j$  as solved.
    - 2.4.3. If  $n_j$  has been marked solved or if its cost has been changed, add to the set  $S$  all the ancestors of  $n_j$ .

The algorithm first expands one node selected by tracing down through the marked connectors. The marks indicate a tentative optimal-partial-solution graph from each node. The next operation is to update the cost of the selected node and to mark the connector through which the minimum is achieved. The new updated cost is propagated backward.

If for each node the estimated cost is a lower bound of the cost of an optimal solution graph rooted at the node, the algorithm is admissible. The proof is given elsewhere (3). An optimal-solution graph is obtained by tracing down from  $n$  a hyperpath through the marked connectors. The cost of the solution is the value of  $h^*(n)$  at the end of the search.

The cost revision in the algorithm becomes simpler by making the additional assumption that the estimate of the cost is consistent. This means that for each node  $n$  and for each outgoing connector the estimated cost associated with the node  $n$  cannot be larger than the sum of the estimated costs of all of the children of  $n$  through that connector plus the cost of the connector itself. With this assumption step 2.4.3 of the algorithm can be modified because the cost of a node can only increase. Not all ancestors of node  $n_j$  need a cost revision, but only those that are connected through a marked connector to a node whose cost has been revised. Step 2.4.3 of the top-down algorithm for heuristic search can be modified as follows:

- 2.4.3. If  $n_j$  has been marked solved or if its cost has been changed, add to the set  $S$  all of those parents of  $n_j$  such that  $n_j$  is one of their output nodes through a marked connector.

The top-down Algorithm presented above with the two additional assumptions is called AO\*. The name was given by Nilsson (6).

Additional algorithms based on heuristic search are described by Pearl (4).

### Use of Problem-Reduction Methods

Some of the early work in AI used problem-reduction techniques. For instance, the logic-theory machine of Newell, Shaw, and Simon (16) proved theorems in propositional calculus (see Logic, propositional) by working backward from the theorem to be proved and by keeping track of alternative paths with a tree of subproblems.

Gelernter (17) worked on a geometry-theorem prover that was based on problem-reduction techniques. Each problem to solve is decomposed into subproblems; every primitive problem is either an axiom or a "given" hypothesis. His program is considered the first program that was able to handle conjunctive subgoals.

Slagle (2) used subproblem trees in his work on symbolic integration. He was the first to call these trees AND/OR trees. Symbolic integration problems are decomposed into subproblems by observing that the integral of a sum can be transformed into the sum of integrals. Since all of the expressions have to be integrated, this decomposition generates AND nodes. Each expression that can be integrated in alternative ways generates OR nodes.

Martelli and Montanari (5) show that dynamic programming can be formulated as AND/OR search. The algorithm for heuristic search in AND/OR graphs that is illustrated above is the same as the algorithm that they describe.

Levi and Sirovich (18) introduce the definition of generalized AND/OR graphs to model subproblem interdependence. The main difference with conventional AND/OR graphs is that they allow reduction operators to reduce simultaneously a set of input problems to a set of output problems. Because of problem interdependence, the search algorithm must be able to find different solutions to the same subproblem. This capability is not needed when all of the subproblems are independent because a particular solution to a subproblem can never prevent finding a solution to another subproblem. They give an admissible algorithm and they prove that generalized AND/OR graphs are equivalent to type 0 grammars. They show that they can handle problems requiring consistent binding of variables and problems involving the use of scarce resources. However, they do not address the much more important issue of subproblems that share variables.

Charniak and McDermott (19) propose a different interpretation of AND/OR trees. They see AND/OR trees as goal trees rather than problem-reduction trees. Each node represents a plan that is partially undefined. Only tip nodes contain fully specified plans. The tips of the tree are usually AND nodes with no children. A node that corresponds to a fully specified plan is called a success node. A node that cannot be satisfied is called a failure node. For the search process the goal tree is transformed into a plan-goal tree. The AND nodes are moved into the partial plans attached to each node. The remaining tree is an OR tree that can be searched with search algorithms used for conventional graphs.

Chang and Slagle (20) describe a different method of using AND/OR graphs that is much more similar to the state-space approach.

AND/OR graphs are important in theorem proving (qv). A theorem prover starts with a set of axioms and a set of inference rules. At each step new statements can be deduced from a

subset of axioms and previously deduced statements. When a conclusion is obtained, the internal order in which the statements were derived is not important as long as they are available at the appropriate time. An AND/OR graph is appropriate to describe the search space. Kowalsky (21) discusses AND/OR graphs in the context of theorem proving. Loveland (22) shows how to use AND/OR trees to represent deduction and search space instead of using resolution.

The production rules used in an expert system (qv) can be represented as an AND/OR graph (23). Each primitive node is a known fact, and each problem is the consequent of a rule. This allows simple graphical representations of rules. The elementary explanation mechanism of rule-based systems (qv) (the famous "how" and "why" questions) is simply obtained by identifying the children or the parent of a node.

### BIBLIOGRAPHY

1. N. J. Nilsson, *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.
2. J. R. Slagle, A Heuristic Program that Solves Symbolic Integration Problems in Freshman Calculus, in E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, pp. 191–203, 1963; also in *JACM* 10, 507–520 (1963).
3. A. Martelli and U. Montanari, Additive AND/OR graphs, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford, CA, pp. 1–11, 1973.
4. J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, Reading, MA, 1984.
5. A. Martelli and U. Montanari, "Optimization decision trees through heuristically guided search," *CACM* 21, 1025–1039 (1978).
6. N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, 1980.
7. P. A. Hall, "Equivalence between AND/OR graphs and context free grammars," *CACM* 16, 444–445 (1973).
8. A. Barr and E. Feigenbaum, *The Handbook of Artificial Intelligence*, Vol. 1, William Kaufman, Los Altos, CA, 1982.
9. S. Amarel, On Representations of Problems of Reasoning about Actions, in D. Michie (ed.), *Machine Intelligence*, Vol. 3, American Elsevier, New York, pp. 131–171, 1968.
10. G. VanderBrug and J. Minker, "State-space, problem-reduction, and theorem proving," *CACM* 18, 107–115 (1975).
11. R. Kowalsky, "Algorithm = logic + control," *CACM* 22, 424–436 (1979).
12. L. M. Pereira, Selective backtracking, in K. Clark and S. Tarnlund (eds.), *Logic Programming*, Academic Press, New York, pp. 107–114, 1982.
13. D. E. Smith and M. R. Genesereth, "Ordering conjunctive queries," *Artif. Intell.* 26, 171–215 (1985).
14. H. A. Simon, *The Sciences of the Artificial*, 2nd ed., MIT Press, Cambridge, MA, 1981.
15. R. Waldinger, Achieving Several Goals Simultaneously, in E. Elcock and D. Michie (eds.), *Machine Intelligence*, Vol. 8, Ellis Horwood, Chichester, UK, pp. 94–136, 1977.
16. A. Newell, J. Shaw, and H. Simon, Empirical Explorations of the Logic Theory Machine, in E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, pp. 109–133, 1963.
17. H. Gelernter, Realization of a Geometry Theorem Proving Machine, in E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, pp. 134–152, 1963.

18. G. Levi and F. Sirovich, "Generalized AND/OR graphs," *Artif. Intell.* **7**, 243–259 (1976).
19. E. Charniak and D. McDermott, *Introduction to Artificial Intelligence*, Addison-Wesley, Reading, MA, 1985.
20. C. L. Chang and J. R. Slagle, "An admissible and optimal algorithm for searching AND/OR graphs," *Artif. Intell.* **2**, 117–128 (1971).
21. R. Kowalski, AND/OR Graphs, Theorem Proving Graphs, and Bi-directional Search, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 7, Edinburgh University Press, Edinburgh, U.K., pp. 167–194, 1972.
22. D. W. Loveland and M. E. Stickel, "A hole in goal trees: Some guidance from resolution theory," *IEEE Trans Comput.* **C-25**, 335–341 (1976).
23. P. H. Winston, *Artificial Intelligence*, Addison-Wesley, Reading, MA, 1984.

J. SLAGLE AND M. GINI  
University of Minnesota

## PROBLEM SOLVING

Problem solving is the central phenomenon studied in AI. It is a process that involves finding, or constructing, a solution to a problem. Such a process is carried out by a problem solver executing a problem-solving procedure. A major goal of AI is to develop and study problem-solving systems that are computationally efficient and that are effective over a broad range of problems.

In this entry the notion of a problem and its formulations for a problem-solving system are presented, and issues of choosing a problem representation are discussed. A broad operational classification of problems is given, and major procedural schemas for problem solving are described. Special categories of problems studied in AI, including problems of planning (qv), learning (qv), theory formation, and analogical reasoning (see Reasoning; Representation, analog), are discussed. Throughout, comments are made about the state of research in this area.

### Declarative Problem Formulations

Posing a problem to a problem-solving system amounts to communicating to the system a problem statement in the following general form:

Given a domain specification  $D$ , find a solution  $x$  such that  $x$  is a member of a set of possible solutions  $X$  and it satisfies the problem conditions  $C$ .

In such a statement it is assumed that both the set  $X$  and the conditions  $C$  are expressed in terms of concepts in  $D$ . The problem-solving system can "understand" the problem statement—and it can then proceed to perform a solution-seeking process in response to the statement—if it has means of interpreting and effectively using the information in  $D$ ,  $X$ , and  $C$ .

This type of problem statement is very close to the concept of a declarative formulation of a problem; it is discussed in detail in Refs. 1 and 2. Also, it is closely related to the notion of a set representation of a problem, introduced by Newell and Simon (3).

The nature of a solution  $x$  depends on the problem at hand.

It can be a number, a truth value, a proof, a transportation schedule, a computer program, a theory within a formal system, etc. The specification of the set  $X$  amounts to a characterization of the possible (candidate) solutions for the problem. Typically, such a characterization is given in terms of necessary conditions for the internal structure of solutions or in terms of rules for constructing legal solutions by aggregating available solution elements in the domain. In many cases of interest to AI, the set  $X$  is specified in the form of a generator for members of the set. The problem conditions  $C$  specify further constraints that a member of  $X$  (a candidate solution) must satisfy in order for it to be accepted as a solution for the problem under consideration.

The domain specification  $D$  is a body of basic knowledge about the problem domain: types of objects and predicates that enter in the specification of the problem, relationships among them, and special characteristics of the problem environment under consideration. In general,  $D$  includes a description of a system of concepts in terms of which specific problems in the domain can be expressed, understood, and processed.

In essence, a problem statement contains a description of the solution object in one form and a request to find a description of the object in another, specified, form. The second form is more directly useful to the poser of the problem. In most cases it consists of an explicit specification of the internal structure of the solution object, and the first form provides an implicit characterization of the object.

A problem statement represents a well-defined problem for a given problem-solving system if there exists an effective procedure, performable by the system, that will determine if an object proposed as a solution is in fact a solution (3). *Performable* is used to mean that a "reasonable" amount of computational effort is needed to perform the test. The notion of a well-defined problem was proposed by McCarthy in 1956 (4). Related notions of well-structured and ill-structured problems are discussed by Simon in Ref. 5. For a problem to be well-defined it is required that its associated  $X$  and  $C$  are "made known" to the problem-solving system that will handle it and that the system should have effective and relatively efficient procedures for determining whether a proposed solution object is a member of  $X$  and whether it satisfies the conditions  $C$ . In general, the problem-solving processes that are developed and studied in AI are concerned with well-defined problems.

Frequently, not all the information needed to state a well-defined problem is explicitly communicated to a problem-solving system. Often, the relevant  $D$  and the set  $X$  may not be given, explicitly or they may be presented in incomplete form during an initial specification of the problem. The problem solver must then complete its "conceptualization" of the problem by inferring the missing information from context or in some other way. This act of selecting, or completing, a domain description for a problem, i.e., of placing the problem in some conceptual frame of reference, represents a crucial decision that can greatly influence the ease with which a solution to the problem can be found. Indeed, under certain conditions, as some relevant psychological studies suggest (6), the selection of alternative frames within which a problem is cast may result in radically different solutions to the problem. It has been well known that the process of "mobilizing" relevant information to complete a problem specification is an important part of the problem-solving process (7). In some cases this problem-acquisition phase, which can be regarded as a "problem-understanding" task, is the central part of the problem-

solving effort. Within AI there has been relatively little work on problem-understanding problems (8). So far, the emphasis of research in this area has been on how to construct some valid problem statement in response to input information about a problem that is presented in natural language and is not guaranteed to be complete. This is an area in which much more work is needed.

Regardless of the specific way in which a problem statement comes about—whether it is formulated all at once or is assembled from fragments, some externally communicated and some “mobilized” internally—it is conceptually useful to assume that there always exists such a statement (or its equivalent, from the point of view of information content and access) at the starting point of any problem-solving process. If the problem-solving activity requires a problem-acquisition stage whose end point is a problem statement that will govern the next stage of solution construction, it is useful to conceive of the situation as consisting of two well-defined problems: a problem-acquisition problem and a solution-construction problem. To date, work in machine problem solving has concentrated mainly on the second part, and the problem-acquisition part was left largely to humans.

It is useful to abstract the main components of a problem statement and to define in their terms the concept of a declarative problem representation (declarative problem formulation). Thus, the declarative representation of a problem  $P$  can be defined as

$$\text{Rep}_d(P) = (D, X, C)$$

Usually, a problem is formulated as a member of a problem class. Such a formulation requires a specification of the class and additional information for distinguishing the given problem from all the others in the class. The importance of the notion of a class of problems lies in the fact that problem-solving systems are commonly designed for problem classes.

The specification of a class of problems  $p$  can be given in terms of a schema of problem conditions, SC, and of a problem data domain PD. The schema SC specifies the type of problem conditions that enter in the formulation of problems that are members of the class; and the domain PD is a set of parameter values, each associated with an individual problem in the class. A problem  $P$  in  $p$  has an associated parameter value  $d$ , which is an element of the domain PD. The condition schema for a problem class has variables (or “slots”) that can be assigned values (can be instantiated) from the domain PD. The problem conditions  $C$  of a problem  $P$  in class  $p$  are obtained by assigning the parameter value  $d$  associated with  $P$  to the variables (slots) of the condition schema for the class. Thus, given the schema SC and the parameter value  $d$  associated with  $P$ , the conditions  $C$  of the problem  $P$  can be uniquely specified. In other words, the conditions that define an individual problem can be expressed as a specialization of the condition schema of a class in which the problem is assumed to belong. In view of these notions, the declarative representation of a problem class  $p$  can be defined as

$$\text{Rep}_d(p) = (D, X, SC, PD)$$

and the declarative representation of a problem  $P$  in terms of a class  $p$  can be defined as

$$\text{Rep}_d(P, p) = (D, X, SC, PD, d)$$

To date, the formulation of a problem in terms of a class is mainly done by people. Research on “problem understanding” (8) and on problem solving by analogy (9) is relevant to the mechanization of this task. Much more work is needed in this area.

### Procedural Problem Formulations

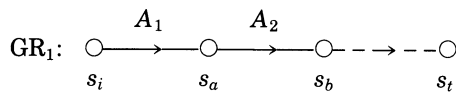
For a problem-solving system to proceed with a solution-seeking activity in response to a problem, it must have available a formulation of the problem in procedural form; i.e., the formulation must be in a form that can be readily assimilated/accepted by a procedural schema for problem solving that is available to the system. The assimilation of a problem formulation into a procedural schema amounts to defining a specific problem-solving procedure. Such a procedure is an instantiation of the schema in accordance with the contents of the procedural formulation.

In order to understand the nature of problem-solving procedures, it is helpful to focus on ways in which solutions are constructed and more specifically on ways in which elements of a problem formulation are used to control the solution-construction process. In many AI tasks the desired solution is an object with considerable internal structure assembled from smaller parts during the problem-solving process. For example, in a problem of reasoning about actions, a solution has the form of a structured plan made of a sequence of actions that are suitably combined to ensure that the plan satisfies desired goals; in a problem of program synthesis (see Automatic programming), a solution has the form of a program made of statements in a programming language that are appropriately articulated to achieve the desired program characteristics. In these types of situations the set of candidate solutions  $X$  can be regarded as a solution language; and the language can be specified by a solution grammar  $G(X)$  (see Ref. 2). Candidate solutions can be represented as graphs of special types. These graphs can be viewed as analogs of strings in ordinary language.

Solution graphs are made of terminal and nonterminal elements. Terminal elements represent well-specified solution fragments. An example of a terminal element is a proposed action to be taken at a particular situation in a planning problem. Nonterminals represent parts of solutions that are incompletely specified; they can be regarded as “gaps” in a solution structure. Typically, the rules of replacement of solution grammars are nondeterministic. This reflects a common situation encountered in AI problem solving: at any one point in the construction of a solution, there are several alternative ways of proceeding with the construction process. In other words, to start filling an “open gap” in a candidate solution, one has the option of using several alternative solution fragments. This element of a priori uncertainty about how to proceed in building a structured object from a given set of building blocks is fundamental to the processes studied in AI. It is also responsible for the combinatorial growth of computational effort needed to explore the construction of solution alternatives.

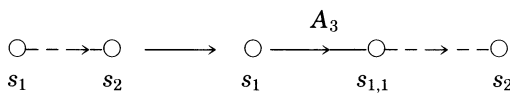
As in the case of ordinary grammars, solution grammars need a designated nonterminal element as a starting element for the generation (or parsing) of solution graphs. Now, a solution graph can be defined constructively in terms of a process of generation that starts with the starting element of the grammar and proceeds by applying a sequence of grammar rules until the solution graph is obtained.

For example, in a problem of finding a minimal sequence of actions that can transform a situation  $s_i$  to another situation  $s_t$  in some domain, the set of candidate solutions  $X$  consists of all "legal" action sequences, i.e., of all sequences in which a component action satisfies all the sequencing constraints imposed on actions in the domain. This set of legal action sequences can be regarded as the language of solutions in the domain, and it can be specified by a grammar  $G(X)$ . A specific problem in this domain can be seen as a member of a class of problems whose condition schema  $SC$  stipulates that the action sequence that transforms a situation  $s_i$  to a situation  $s_t$  should be minimal. Furthermore, the data domain  $PD$  for the class consists of all the possible values for pairs of situations  $(s_i, s_t)$ . The schema  $SC$ , together with a specific value  $d$  for a pair of situations, define the conditions  $C$  for a specific problem in the class. Solutions and partly specified solutions to problems in this class can be represented by special solution graphs. For example, the following graph  $GR_1$  represents a partly specified solution with only the first two actions  $A_1, A_2$  specified:

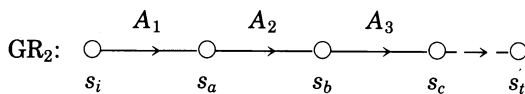


The two segments of the graph from the left represent terminal elements, and the right segment represents a nonterminal. The nodes  $s_a, s_b$  represent intermediate situations obtained from  $s_i$  by applying the actions  $A_1$ , and  $A_2$  in succession. Further specification (construction) of the solution can proceed by applying a rule replacement of the grammar to the graph.

A grammar rule is typically specified in terms of a transition, which has the form



and a set of applicability conditions  $AC$ . The rule is applicable to a partially specified solution, such as the one represented by  $GR_1$ , if the left side of its transition matches the nonterminal in  $GR_1$  (i.e., if  $s_b$  matches  $s_1$  and if  $s_t$  matches  $s_2$ ) and if the conditions  $AC$  are satisfied by  $GR_1$ . If the rule is found to be applicable, it can be applied, thus producing a new graph  $GR_2$ :



which represents an extension of the previous partly specified solution. The new node  $s_c$  in  $GR_2$  represents the situation resulting when action  $A_3$  is applied at  $s_b$ ; it corresponds to  $s_{1.1}$  in the right side of the rule's transition, with appropriate bindings of variables. Note that the applicability conditions  $AC$  that enter in the specification of the solution grammar embody the conditions that govern the composition of individual actions into sequences. These can be viewed as local structural conditions that solutions must satisfy, and they are distinct from the problem class conditions  $SC$  (or the problem conditions  $C$ ) that impose more global constraints on solutions.

A solution grammar provides a model for a generator of candidate solutions that can be used in a problem-solving system. At any intermediate stage of its problem-solving activity, a system with such a generator will have under consideration

several alternative solution candidates in different stages of construction. In order to decide on which solution candidate the activity of the generator should focus attention next and which generation action (grammar rule) to select in continuing the construction of a solution candidate, the system must have an appropriate controller. Such a controller would operate under the guidance of control knowledge  $CK$ . One part of  $CK$  would consist of domain-dependent heuristic rules (see Heuristics) for attention control and action selection, and another part would consist of general procedural knowledge that is relevant to the operation of the problem-solving system.

The conditions  $C$  that enter in the formulation of a problem can be used to influence solution construction activities in two ways: by a posteriori testing whether a candidate solution satisfies the problem conditions (and, possibly, evaluating a candidate in terms of the degree to which it satisfies the conditions) and by a priori constraining the generator (and possibly influencing the controller) so as to minimize the chances of working on solution candidates that do not satisfy the problem conditions. Thus,  $C$  can be used to test candidates proposed by  $G(X)$  and also to influence the generation of candidates via constraints imposed on  $G(X)$  and  $CK$ . Let  $POST(C)$  be a procedure for testing whether a solution candidate satisfies  $C$ , and let  $PRIOR(C)$  be a procedure for a priori constraining the solution generator (and possibly the controller) in a manner that minimizes the production of solution candidates that are inconsistent with  $C$ . In general, given  $C$ , it is fairly straightforward to specify  $POST(C)$ , but specifying a good procedure  $PRIOR(C)$  can be a demanding task for certain types of problems.

The procedural schemas used in AI specify solution-construction processes that involve generators and controllers in various combinations as well as procedures  $PRIOR(C)$  and  $POST(C)$  for using problem conditions in a priori and a posteriori modes. The most general (and weakest) procedural schema is the generation-and-test schema. More specialized and refined schemas, that are in the general mold of generate and test but that are each "best suited" to handle different types of problems, have been developed in AI, and they are discussed below.

Consider a problem  $P$  with a declarative formulation  $(D, X, C)$ , that is being solved in accordance with the generate-and-test schema. A system operating in accordance with the schema proceeds in a sequence of cycles, and it involves the following information entities:

1. a domain database that implements  $D$ ;
2. a generator based on some grammar  $G(X)$ ;
3. a controller governed by control knowledge  $CK$  (which may depend on  $C$ );
4. a procedure  $PRIOR(C)$  for using the problem conditions  $C$  in an a priori mode to control  $G(X)$  and  $CK$ ;
5. a procedure  $POST(C)$  for using  $C$  in an a posteriori mode to test and/or evaluate generated solution candidates; and
6. a working database where a record of solution construction activities is kept.

The basic operation of the generate-and-test schema is as follows:

*Initialize.* Set the characteristics of  $G(X)$  and  $CK$  that depend on  $PRIOR(C)$ ; direct  $G(X)$  to perform an initial solution-construction action.

*Test.* Examine, via  $\text{POST}(C)$ , whether the working database contains a desired solution; if yes, exit with success; otherwise, continue.

*Generate.* Direct  $G(X)$  to perform a solution-construction action based on the current state of the working database and in view of  $CK$ ; if no action is possible or if available computational resources are exhausted, exit with failure; else, go back to the test step.

Various other procedural schemas differ in several ways from generate and test; however, they all involve the same types of information entities in their operation. Thus, we can define the procedural representation (or procedural formulation) of a problem  $P$  as follows:

$$\text{Rep}_p(P) = (D, G(X), \text{PRIOR}(C), \text{POST}(C), CK)$$

By using information in  $\text{Rep}_p(P)$  to specify parameters (or slots) of a problem-solving schema, the schema can be transformed into a specific procedure for solving  $P$ .

The procedural representation of a problem  $P$  in terms of a class  $p$  can be defined as

$$\text{Rep}_p(P, p) = (D, G(X), \text{PRIOR}'(\text{SC}, d), \text{POST}'(\text{SC}, d), CK, \text{PD}, d)$$

where  $\text{PD}$  stands for the problem-data domain of the class,  $d$  is the specific problem data associated with  $P$ , and  $\text{SC}$  represents the schema of problem conditions for the class  $p$ . The problem conditions  $C$  of an individual problem  $P$  are defined as the instantiation of  $\text{SC}$  by the problem-data  $d$ . The procedure  $\text{PRIOR}'(\text{SC}, d)$  is functionally equivalent to  $\text{PRIOR}(C)$ ; similarly,  $\text{POST}'(\text{SC}, d)$  is functionally equivalent to  $\text{POST}(C)$ .

Currently, the procedural formulation of a problem for a problem-solving system that is organized in accordance with a given schema is done almost exclusively by people. This activity can be seen as programming the problem for an AI system—at a very high level of specification.

### The Problem of Representation in Problem Solving

The choice of a problem formulation for a given problem-solving schema strongly affects the ease (complexity) of constructing a solution for the problem. Understanding principles for making such a choice and finding ways of mechanizing (parts of) the choice are at the heart of the problem of representation in problem solving (see Refs. 1, 2, and 10–13). One of the issues in this area is how to transform a declarative problem formulation into a procedural formulation that would be appropriate for a given problem-solving schema.

The choice of a specific representational framework for  $D$  is a significant component of the problem of representation. Not only should such a framework provide sufficient linguistic and control facilities for expressing and processing domain knowledge of relevance to the problem on hand but it must also do so in an efficient manner. Essentially, here is the problem of choosing an appropriate “view of the world” (in the form of a theory of some sort, or of a model) within which a given problem is to be handled, where “appropriateness” refers to questions of problem-solving efficiency.

There is no clear a priori basis for partitioning knowledge between  $D$  and  $G(X)$ . It is conceivable that rules for generating (or for characterizing) solutions in a domain might be included in a single body of knowledge about the domain. This

has been the practice in many of the AI systems developed in recent years, where general information about a domain and about possible solution-construction actions in the domain are embodied in a single knowledge base of the system. However, the distinction between  $D$  and the specification of  $X$  is conceptually useful in the context of problem-representation choices. It allows separate consideration of the choice of a “view of the world” (i.e., of a domain conceptualization) and the choice of specific rules for constructing solutions to classes of problems within that world. The rules that are used to characterize the set of candidate solutions embody knowledge about a specific class of problems in the domain; and the capturing and expressing of such knowledge depends heavily on the domain conceptualization. For a given domain of discourse represented by a given  $D$ , there may be several classes of problems, each having its own grammar for solutions of problems in the class. A central issue in the problem of representation is how to choose the most appropriate solution grammar for a specific class of problems in a domain.

Experience in the area of problem representations shows that mechanizing the process of choosing a “good” initial problem formulation is a very difficult task. Recent work shows that a more promising approach is to assume that one or more problem formulations are somehow available and to explore processes of reformulation, in particular those that lead to reformulations that result in increased efficiency of problem solving (2). In order to achieve “good” reformulations, it is essential to have methods for acquiring additional domain-specific knowledge and of using it “appropriately” in the restructuring of key components of a formulation, in particular, of the solution-grammar rules, which determine ways of articulating solution structures from substructures, and of the procedures  $\text{PRIOR}$  and  $\text{POST}$ , which determine how problem conditions influence solution construction. To mechanize processes of reformulation, progress is needed in theory formation and program synthesis.

In recent years considerable progress was made in the area of knowledge representation (qv). Work in this area is relevant to issues of problem representation. Typical questions of knowledge representation are how to encode various objects, facts, rules, and concepts of a domain in “appropriate” data structures, i.e., in data structures that can be created and/or updated and accessed in ways that are computationally efficient. Propositional representations, production rules, and various types of “structured objects” are examples of approaches that have been developed and studied in this area. The availability of good knowledge representations is a necessary condition for good problem representations; however, it is not a sufficient condition. What is needed in addition are good choices of a grain of description and of knowledge distribution over the various parts of a problem-solving system, where “good” is meant in the sense of leading to effective and efficient ways of handling problems (in a given class) by the system.

### Solution Methods. Problem-Solving Procedures

In general, a computer-based problem-solving procedure is designed to handle a given class of problems. The input domain of the procedure corresponds to the data domain of the problem class, and a specific element of the input domain is representative of a specific problem in the class. If a procedure is executed for an element of its input domain, which stands for a given problem, a process is carried out that either terminates with



"success" and returns a solution to the problem or "failure," owing, for example, to errors or to exhaustion of available computational resources. A procedure expresses a method of solution for a problem class in terms of statements that are interpretable and executable by the system according to a well-defined plan. The procedure can be seen as embodying a specific way of utilizing knowledge about a problem class (which is contained in the problem class formulation) that results in a transformation of the initial problem class into a configuration of "simpler" problem classes that the system "knows how to solve."

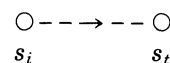
The success of "conventional" problem solving by computer in many areas of science, engineering, and data processing owes a great deal to the large amounts of systematic knowledge that exist in these areas and to the availability of strong methods of solution finding that use the available knowledge in a highly effective manner. Under these conditions, efficient problem-solving procedures, whose correctness and optimality can often be justified on formal grounds, can be custom-produced. The situation is quite different in AI, where the emphasis is on problems with limited amounts of systematic knowledge and where solution-finding processes rely primarily on weak methods (3,14,15). These are general solution-seeking methods, such as generate and test, that are widely applicable over families of problem classes. Under the guidance of heuristic knowledge, the weak methods use available knowledge about a problem in certain generic ways in their attempt to search for a solution. The concept of heuristically guided search is central to the problem-solving processes studied in AI. In general, there is no sharp dividing line between the problem-solving procedures of AI and the procedures used in "conventional" computing. The distinctions are based on the relative amounts of systematic vs. heuristic knowledge available to the procedures, and on the form in which the available knowledge is used (see Refs. 1 and 16).

### Derivation-Formation Spectrum of Problems

Experience shows that problem-solving power depends on the degree to which problem conditions can be made to influence directly the process of solution construction. Furthermore, the amount of influence exerted by problem conditions depends on their relative use in an a priori or an a posteriori mode. The more dominant the a priori mode, the more powerful (more selective, more sure) the problem-solving process. By focusing on the different ways in which problem conditions can be used to control solution-construction processes, one is led to a broad classification of problems along a spectrum. Problems are ordered in this spectrum by the degree to which their problem conditions can be used to control directly the solution-generation process. At the two ends of the spectrum there are derivation problems and formation problems (see Refs. 1 and 12). These two families of problems correspond, respectively, to the problems to prove and problems to find discussed by Polya (7).

**Derivation Problems.** In problems of derivation type, the situation can be viewed as follows: The problem conditions are given as parts of the desired solution structure, and the task of the problem solver is to complete this (partially specified) structure by using rules for solution construction that are specified by a given solution grammar—in such a manner that the initially given parts are well integrated in the structure. Usually, the problem conditions specify boundary parts of the

solution structure, and the problem-solving process consists of finding a "connecting bridge" between the given boundaries by piecing together solution elements in accordance with the rules of the grammar. It is characteristic of a derivation problem that the solution-construction process is strongly controlled by the problem conditions. Problem conditions are primarily used in an a priori mode in the process of solution construction. This can be seen by examining the way in which procedures PRIOR(C) and POST(C) appear in procedural formulations of derivation problems. In these problems, a procedure PRIOR(C) can be seen as specifying the starting element of the solution grammar for the problem; thus, the problem conditions  $C$  enter directly in this specification. For example, in a problem of finding a sequence of actions that can take one from a situation  $s_i$  to another situation  $s_t$ , the starting element of the grammar is set [by PRIOR(C)] as follows:



A procedure POST(C) can be seen as testing whether a candidate solution is completely specified, i.e., if the solution graph is made exclusively of terminal elements; if yes, it is an acceptable solution, otherwise, the procedure (if appropriately "informed") assigns to the incomplete solution some estimate of "distance" to completion.

A typical example of a derivation problem is the task of constructing a proof to a theorem in a formal system (see Theorem proving). Other examples are problems of path finding (between well-specified points) (see Path planning and obstacle avoidance) and problems of plan generation (to satisfy explicit goals) (see Planning).

**Formation Problems.** In problems of formation type, the relationship between problem conditions and the structure of a solution is more complex than in derivation problems. Here the problem conditions are given in the form of properties that the solution as a whole must satisfy, and the problem solver is to generate a solution description within a language of solution structures that satisfies the required properties. Typically, no choice of solution elements can be determined directly from the given problem conditions. The solution process cannot proceed by reasoning from the problem conditions to specific parts of the solution, as is possible in derivation problems. The general approach here is to generate candidate solutions in the language of solutions and to test them against the problem conditions. The problem solver must have an effective procedure that can take as input the description of a candidate solution and can decide whether (or how well) the candidate satisfies the given solution properties. An entire solution candidate must be assembled before it can be tested. Problem conditions are primarily used in an a posteriori mode in solution construction. This can be seen by examining the way in which procedures PRIOR(C) and POST(C) enter in procedural formulations of formation problems.

As in the case of derivation problems, PRIOR(C) can be regarded as specifying the starting element of a solution grammar  $G(X)$  for the problem. However, in the present case the problem conditions  $C$  enter indirectly and weakly in the specification. Consider, for example, the problem of synthesizing a program in a given language  $L$  to satisfy a set of I/O correspondences, where the inputs are of data type  $T_1$  and the outputs of data type  $T_2$ . Here, the desired solution is a program in the language  $L$ ; and  $L$  is specified in terms of a program grammar

$G(L)$ . The set of I/O correspondences define the problem conditions  $C$ . In the present problem PRIOR( $C$ ) may simply specify the starting element of  $G(L)$  in the form of a program schema that satisfies certain properties of the set of correspondences, e.g., the starting schema has inputs of type  $T_1$  and outputs of type  $T_2$ . By successively applying rules of replacement from  $G(L)$  to the starting element, the starting program schema becomes increasingly better specified until a completely specified candidate program is generated.

A procedure POST( $C$ ) in the formulation of a formation problem can be seen as doing the following: It takes a completely specified candidate solution, and it tests whether it satisfies the problem conditions  $C$ ; if yes, it is an acceptable solution, otherwise, the procedure (if appropriately "informed") assigns to the candidate solution an estimate of "degree of success" in satisfying  $C$ . Testing whether a candidate satisfies the given problem conditions involves computing properties of the candidate so that a direct assessment can be made, via comparison, of whether the candidate satisfies the conditions. In some cases this process of testing can be a complex problem-solving effort by itself. In the program-synthesis example a candidate program is tested by running it over all the given inputs and by comparing the computed outputs with the given outputs. Many problems in theory formation and in the interpretation of experimental data are of the formation type. In these types of situations testing a candidate theory may involve deducing consequences of the theory and comparing them with experimental data.

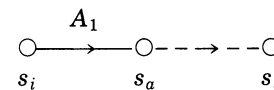
### Major Problem-Solving Schemas

The bulk of work on problem-solving processes in AI has been done in the area of derivation problems. Two problem-solving schemas have been extensively used and studied in this area: the production schema (or state-space search method) and the reduction schema (see Refs. 1 and 17). A third schema, called relaxed reduction (2), which is especially appropriate for goal-directed reasoning in planning problems, has received increased attention in recent years. Each of these schemas represents a different approach to constructing a solution for a problem of derivation type, and it has different ways of utilizing knowledge about a problem in controlling the solution-construction process.

**Production Schema.** The main concepts that enter in the specification of a production schema are states, moves, an evaluation function for states, a move-selection function, and a state-selection function. A state represents the current state of a solution candidate, from the point of view of the problem-solving process. More specifically, the state must provide a description of the current status of the candidate under construction that is sufficient for determining whether the candidate is already a solution or, if not, for deciding what move to apply on the candidate in order to continue the solution-construction process. Applying a move represents a solution-construction action. A move is characterized by its effect on a (partly specified) solution candidate and by its applicability conditions. The effect of a move can be represented by a transition between a "from" state and a "to" state. The applicability conditions of a move specify under what conditions the move can be applied from a state. A move-selection function chooses one or more of the applicable moves for actual application. A state-evaluation function assigns to a state (which represents

an incompletely specified solution candidate) a heuristic estimate of "effort" needed to reach a completely specified solution from the state. Such an estimate may be in the form of a number of moves needed to complete a solution from the given state. This can be seen as the distance between the current state and the nearest state representing a completely specified solution. Alternatively, the estimate may be interpreted as a measure of computational effort needed to obtain a solution from the state. A state-selection function chooses the state on which the problem-solving process must focus attention next, on the basis of the values assigned to states by the evaluation function.

Commonly, a solution has the form of a sequence of situations starting from a given initial situation and ending with a given terminal situation (or with a situation from a given set of terminal situations); each of the situations in the sequence (except the first) is obtainable from the previous one by application of a permissible action (or operator) in the domain. The notion of a situation is different from the notion of a state, but it is closely related to it. Although a situation portrays a "state of the world," a state represents a "state of reasoning about states of the world." For example, in a problem of reasoning about actions, where the task is to find a sequence of actions that can transform a situation  $s_i$  into a situation  $s_t$ , one may have a partly specified solution candidate in the following form:



The state (of solution)  $S$  in this case may be defined as the pair  $(s_a, s_t)$ . This concept of state is chosen with the intent of providing all of the information that is necessary for deciding about the next step of solution construction from the state.

There is a close correspondence between the main components of a production procedure and the components of a procedural formulation of a problem. The solution grammar  $G(X)$  together with PRIOR( $C$ ) determine the set of states  $\{S\}$ , the set of moves  $\{M\}$ , and the initial state  $S_0$ . The procedure POST( $C$ ) determines the state-evaluation function. The move-selection and state-selection functions are major parts of the control knowledge CK that define the controller. Thus, it is straightforward to go from a procedural formulation of a problem to the specification of a production procedure for solving it. The value of a procedural formulation lies in the convenience that it offers in specifying key elements of a procedure at an "appropriately high" level of description without having to go into implementation details.

The set of all situations in a domain together with the set of all permissible actions (operators) define a space of situations. This is commonly called a state space, under an interpretation of "state" to mean state of the world. The problem-solving activity carried out by a production procedure can be seen as search (qv) in this space. The search paradigm involving a space of objects that are interpreted as possible states of the world of a problem domain has been dominant in much AI work on problem solving since the early days of the field (3,18).

For a given problem the production procedure can be seen as searching for a solution, starting from a boundary situation (typically, the initial situation) and piecing together in an incremental manner a sequence of actions (operators) until the second boundary is reached. Typically, several actions are applicable to a situation, and this gives rise to a search tree

(more generally, to a search graph) in situation space. The nodes of the tree represent situations, and the branches represent action applications that transform a situation into another. Different approaches to search may be represented by different ways of moving in situation space. It is possible to initiate the search from the terminal situation and to proceed (by inverse application of actions) in the direction of the initial situation. It is also possible to proceed with bidirectional search (qv), where search proceeds simultaneously from both boundary situations toward the middle.

A production procedure "grows" in its working database a tree that is isomorphic with the search tree in situation space; it can be seen as a search tree in state space. The notion of "state" is used here in the sense of "solution state," i.e., as a state of the problem-solving activity. The nodes of the search tree in state space are solution states, and the branches represent move applications that take a solution state into another. This tree represents the alternative solution-construction activities and their relationships during a problem-solving process. An example of search trees in situation space and state space for a simple problem of reasoning about actions is shown in Figure 1. The problem is to find a sequence of actions that takes a situation  $s_i$  to a situation  $s_t$ .

The basic operation of a production schema is as follows:

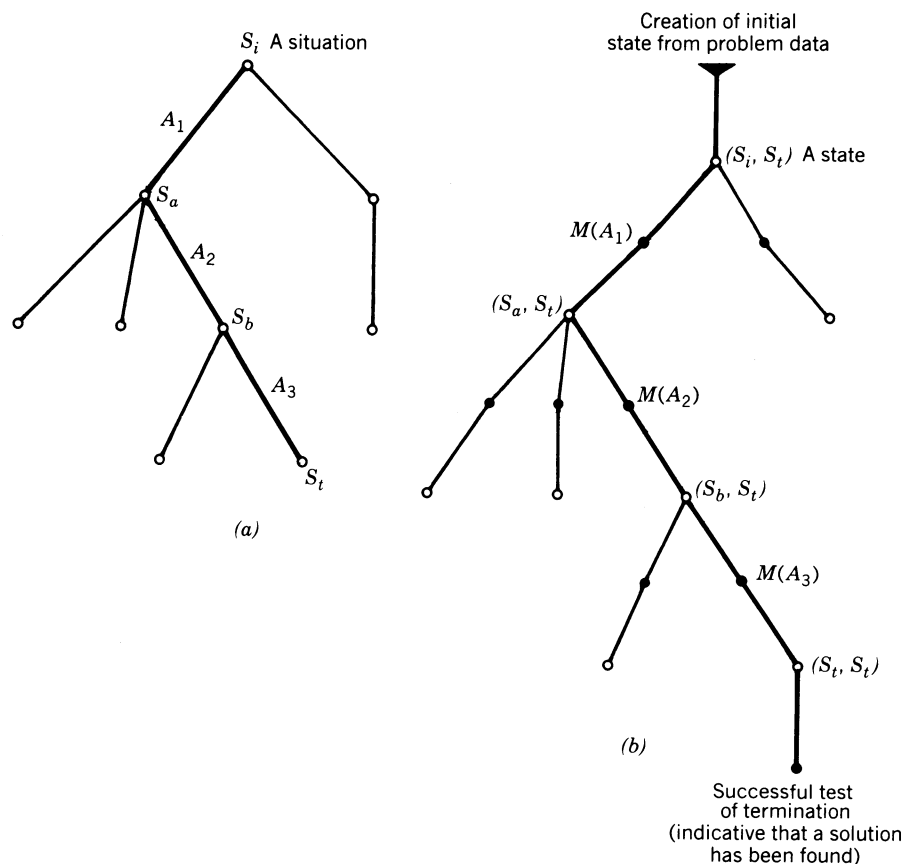
**Initialize.** Set the initial state  $S_0$  from the problem data; make it the root node of the search tree; apply move selection function on  $S_0$  and generate descendant states.

**Test.** Examine if the search tree contains a solution; if yes, exit with success; otherwise, assign evaluation function to new states.

**Generate.**

- If computational resources are exhausted or if there are no unprocessed states in the search tree, exit with failure; otherwise, continue.
- Apply state selection function on the states of the search tree.
- Apply move selection function on selected state.
- Generate new states by applying the selected moves.
- Go to (b).

By choosing different state-selection and move-selection functions, it is possible to implement different search disciplines. Breadth-first search, depth-first search (qv), and best-first search (qv) are disciplines that have been widely used in AI. To each of these disciplines there corresponds a procedure from the class of production procedures. Best-first search procedures choose a state (on which attention is to focus next) on the basis of the best value of a heuristic estimate of "distance to solution" that is associated with states in the search tree. Several algorithms, including A\* (see A\* algorithm), have been developed and analyzed in this area (17,19,20). The performance of these algorithms depends strongly on the quality of the state-evaluation function, i.e., on the accuracy of heuristic estimates of "distance." One of the important problems in this area is



**Figure 1.** Search trees associated with a production scheme. (a) Search tree in situation space (solution is the darkened path from  $s_i$  to  $s_t$ ). (b) Search tree in state space (solution construction trace is represented by the darkened path, dark nodes represent move applications,  $M(A)$  stands for a move corresponding to application of action  $A$ ).

how to choose and improve the evaluation function via acquisition and proper use of domain knowledge (20).

**Reduction Schema.** The main concepts that enter in the specification of a procedure organized in accordance with a reduction schema (see Problem reduction) are very close to those that enter in the specification of a production procedure. The main distinction is in the specification of moves. In a reduction procedure there are two kinds of moves: a set of reduction moves (at least some of which) that transform a state into two or more states that are presumed easier to handle than the original and a set of terminal moves that completely resolve whatever is problematic in a state, i.e., they recognize that a state is solved or they assign an appropriate (known) solution to the state. As in the case of production procedures, the notion of state refers to "state of solution," but it is more circumscribed and local. Suppose that a solution candidate is being constructed and there are two or more "gaps" in the solution structure that need attention. Suppose further that the gaps can be attended independently, and solution construction can proceed for each gap on the basis of characteristic information about the gap. The information about a gap in a solution structure that permits decisions to be made about "how to fill the gap" is precisely the information that enters in the definition of a state of a reduction procedure. The state can be seen as representing such a gap; more generally, it can be seen as representing a problematic situation. Commonly, a state  $S$  has the form of a pair  $(s_1, s_2)$ , where each element of the pair can be seen as describing one side of the gap, which is represented by  $S$ . Often, the elements of the pair can be interpreted as situations, or "states of the world," in the sense used in production procedures. Other interpretations are possible according to the nature of solutions and their gaps. There is a logical interpretation of the concept of state that associates with the state  $S = (s_1, s_2)$  a proposition  $(s_1 \Rightarrow s_2)$  which means that  $s_2$  is attainable from  $s_1$  (in the space where  $s_1, s_2$  are defined). This is equivalent to the proposition that the (problematic) state  $S$  is solvable. These logical interpretations are useful for clarifying the nature of reduction procedures. The principal activity of these procedures can be seen as reasoning about problematic situations.

A reduction procedure starts by setting an initial state, or problematic situation, and its goal is to resolve it by building a nested sequence of reductions obtained by the application of reduction moves that will lead to "simpler" problematic situations, all of which are directly resolvable by terminal moves. The reduction schema reflects a "divide-and-conquer" method of handling a problem. A problem is decomposed into parts, each part is handled separately, and the results of the individual processes are combined appropriately. This can be seen also as representing an approach to constructing a solution that is different from the approach taken in the production schema where a solution is built incrementally and sequentially from one boundary toward the other. In the reduction approach different parts of the solution can be built simultaneously and independently of each other and then combined in an appropriate manner. If "good" reduction moves can be found in a domain, it is possible to increase appreciably the power of problem solving in the domain.

A reduction move that transforms a state  $S_0$  into a set of reduced states  $S_1, S_2, \dots, S_m$  (for  $m > 1$ ) must satisfy at least the logical condition that if  $S_1, S_2, \dots, S_m$  are all solvable,  $S_0$

is also solvable. In addition, a "good" reduction move must reduce the amount of effort needed to solve the original problem. Usually, such a move incorporates a considerable amount of knowledge about the space in which a search for solution takes place. This knowledge is needed in order to determine decompositions of problematic situations into noninteracting, independent, parts.

A reduction procedure "grows" in its working database a search tree with two types of nodes, state nodes and move nodes. Since several moves may be applicable at a state, there may be several descendants below a state node. Since each reduction move is characterized by a specific number of descendant states, a reduction move node will have its characteristic number of descendants in the search tree. A solution is represented in the search tree as a subtree rooted at the initial state and ending with terminal move nodes, where each state node in the tree has exactly one descendant and each move node has its characteristic number of descendants. The state nodes in the search tree are considered to be OR nodes, and the move nodes with more than one descendant are AND nodes. This comes from the fact that a state is solvable if any of its descendant moves lead to a solution, and a reduction move establishes that its parent state in the search tree is solvable only if all its descendant states are solvable. Thus, search trees "grown" by reduction procedures are usually AND/OR trees (see AND/OR graphs). An example of a search tree associated with a reduction procedure is shown in Figure 2.

The basic operation of a reduction schema (i.e., the overall control structure) is basically the same as that of a production schema. However, because of the different nature of states and

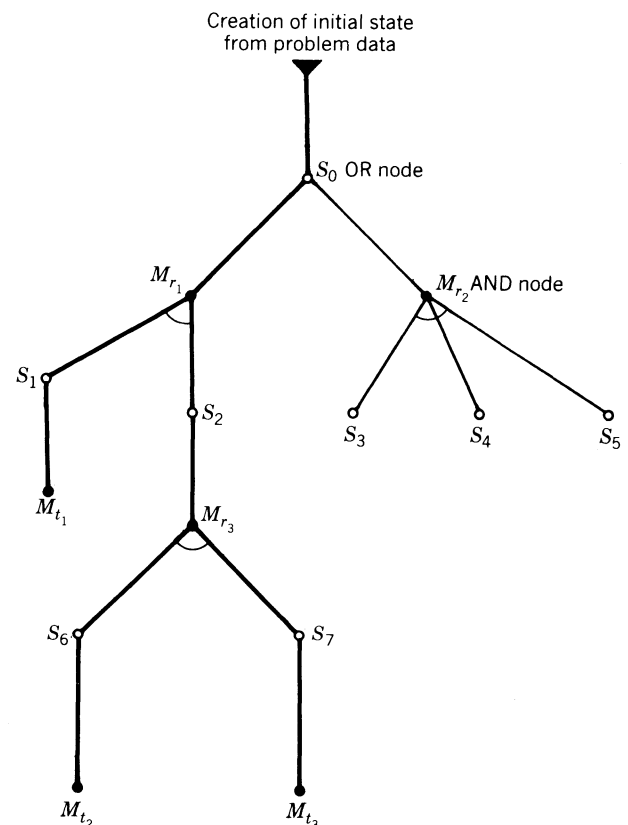


Figure 2. Search tree in state space of a reduction procedure.

moves in the two schemas, the specification of state-evaluation functions and of state-selection and move-selection functions must be based on different considerations. There has not been as much work on the design and analysis of these functions in reduction procedures as in production procedures.

An interesting historical perspective on production and reduction approaches to problem solving is presented in Ref. 21. The reduction approach is close to the concept of analysis developed by the ancient Greeks, and the production approach is related to the ancients' concept of synthesis. In a solution by analysis, work concentrates on conditions of a problem; the conditions are manipulated until one or more points are reached that are known or can be proved to be true. In a solution by synthesis, work moves systematically from a given body of knowledge toward a desired result. Reduction processes are often used in problems where one or more goals are to be achieved (or where the problem conditions can be seen as requiring the achievement of a set of explicit goals). In these problems, states have the form  $(s \Rightarrow g_1 \wedge g_2 \wedge \dots \wedge g_n)$  where each  $g$  represents a goal, and  $s$  represents a "state of affairs" from which the goals are to be achieved. The nature and interdependence between the goals are of crucial importance for the choice of problem-solving method. In certain relatively simple situations it is possible to specify reduction processes for solving such problems. One of the earliest systems for solving such goal-attainment problems via reduction processes was GPS (22).

A key method used in GPS was means-ends analysis. To each "end," which was defined as a difference between a given state of affairs and a desired goal, there was associated a "means" in the form of an operator whose application was expected to reduce the difference. A GPS formulation of a problem included the specification of a "table of differences," which defined these associations for the problem domain. The main approach was to focus on the "most difficult" difference of the current problematic situation and to reduce the problematic situation into the following sequence of (hopefully simpler) problematic situations: starting from the present state of affairs, achieve the preconditions needed for applying an operator that would remove the difference; apply the operator; and work on the remaining problematic situation, which consists of the previous situation with the "most difficult" difference removed. GPS could handle a problem provided that an "appropriate" ordering of the goals was provided. Recently, the GPS approach was generalized to handle more complex goal configurations via the introduction of certain types of macro operators (23).

### Planning Problems, Relaxed Reduction

It is typical of planning (qv) problems that they require the attainment of multiple goals. Many of these problems have as their objective the finding of a structured assembly of actions (usually, a sequence) that obey certain local constraints and that bring about the attainment of a set of goals. In many cases these problems are approached by a GPS-like system. More generally, they are approached by goal-directed reasoning that attempts, to the extent possible, to proceed via reduction of a problem into subproblems. If a reduction into independent subproblems is possible, the conventional reduction schema can be used. However, one is frequently faced with sets of goals that are interdependent. In these types of situations it

is difficult to reason about the choice of a solution construction move from the analysis of an individual goal in the set. A promising approach in this area is offered by the relaxed-reduction schema, which is exemplified by the NOAH system (24). In this approach work on the problem starts by relaxing interactions between goals, focusing attention on the individual goals and working on each of them to a certain depth, as if they were independent. This initial stage is followed by a second stage in which the neglected dependencies are taken into consideration in an attempt to combine the component solution fragments that were developed for each goal in the first stage. The problem-solving activity is made of consecutive cycles, each consisting of this two-stage sequence. Clearly, different types of dependencies between goals require different processes of critique/adjustment in the second stage of a problem-solving cycle. This is an area where much more work is needed.

Many problems of design have similar characteristics with planning problems. In the case of design a functional requirement of a structure to be designed from given structural elements can be seen as the analog of a goal to be attained by a plan made by a sequence of actions. Thus, relaxed-reduction schemas provide promising approaches to the solution of design problems. Again, the specific characteristics of each design problem have a substantial bearing on the definition of the adjustment stage of relaxed reduction, where dependencies that are initially neglected have to be taken into consideration.

GPS was an early attempt by AI researchers at Carnegie-Mellon University (CMU) to develop a general architecture for problem-solving procedures. Interest in this area has grown again at CMU in recent years in connection with the SOAR project (15). The general objective of SOAR is to define a broad schema of problem solving that could be made to specialize to appropriate procedures of different types in response to various problem formulations. Aside from its utility as a system development tool/environment, such a schema has great value for a better understanding of problem-solving processes and of problem formulations.

### Problems with Hybrid Derivation and Formation Characteristics

Most "real-life" problems that are being studied in AI can be placed in some intermediate position between derivation and formation problems. The position in the spectrum depends on the amount of knowledge available about the relationship between solution structures and problem conditions and also on the form of this knowledge. A high proportion of knowledge in the form of links (rules, mappings) from problem conditions to solution structures brings one closer to derivation problems. When the available knowledge is mainly in the form of links from candidate solutions to problem conditions, one is close to formation problems.

**Interpretation Problems.** An important example of a family of real-life problems that is spread over the derivation-formation spectrum (depending on the ways in which their problem conditions can be used) are problems of interpretation. These problems have received considerable attention in recent years in the context of work on AI applications. Typically, a problem of interpretation involves the following kind of reasoning. Given input observations (data) about an individual case in

some domain and a body of domain knowledge in terms of which the case is to be interpreted (understood, explained), the problem is to construct a hypothesis about the input case in terms of the given body of knowledge. The input data are presented in a language of observations that is not necessarily the same as the language of hypotheses. Problems that lie close to the "formation" end of the problem spectrum include in their formulation a set of rules (or a procedure) that can be used to compute consequences of a candidate hypothesis in the language of observations (i.e., expected manifestations of the hypothesis in the world of the observables). Usually, the formulation of an interpretation problem includes, in addition, another set of rules that can be used to reason from input data to candidate hypotheses. Such rules can be used either to evoke the generation of promising hypotheses or to block consideration of unpromising hypotheses. Depending on the strength (selectivity) of this inferential link from data to hypotheses, interpretation problems occupy different positions in the derivation-formation spectrum. The stronger this link, the closer is the problem to the derivation end of the spectrum. Examples of AI systems that have been developed to handle interpretation problems are Heuristic DENDRAL (25,26) and various expert systems for medical diagnosis (27). Although the problems handled by DENDRAL have been close to the formation end of the problem spectrum, most of the medical-diagnosis systems have been close to the derivation end (see Medical advice systems).

The task of Heuristic DENDRAL is to find plausible structures for organic molecules given analytic instrument data from a mass spectrometer and a nuclear magnetic resonance spectrometer and user-supplied constraints on the answers derived from any other source of knowledge available to the user. The system has a generator for producing all the topologically legal candidate structures; this defines a set of possible hypotheses, i.e., the set of possible solutions. A "planning" process analyzes the data and produces constraints that are used to control hypothesis generation so that only candidates that satisfy all the constraints are produced. These candidates are then tested by computing for each of them (on the basis of the model of the analytical instruments) the instrument data that would result if they were analyzed by these instruments and by comparing the computed data with the actual input data. The output of the system consists of the set of candidate hypotheses that score high in this test.

In DENDRAL the generator of legal candidate structures corresponds to the grammar of solutions in a procedural formulation of the task, and the instrument data augmented by user-supplied constraints correspond to the problem conditions. The component POST(C) in such a formulation corresponds to the process of computing the instrument data that would be obtained from a candidate structure and comparing them to the given instrument data. One of the most interesting parts of the system is the "planning" process, which corresponds to the component PRIOR(C) of the procedural formulation. This process consists of analyzing/manipulating the instrument data in accordance with a set of rules and obtaining constraints on the structures and tailoring the structure-generating process to be consistent with the constraints. The problem-solving scheme used in DENDRAL is a modified generate and test. A key part is played by the planning process that controls the characteristics of the generator. All candidate solutions up to a given level of complexity that are consistent with the constraints of the planner are generated and

tested against the instrument data. There is no feedback from test results to the generation process. Finally, all candidates that satisfy the test within given bounds of accuracy are considered to be possible solutions.

**Constraint-Satisfaction Problems.** An important class of problems that is conceptually close to DENDRAL (from the viewpoint of problem-solving method) is the class of constraint-satisfaction (qv) problems. Problems of this class have the following form: Given a set of variables, each to be instantiated in a finite domain, and a set of predicates to be satisfied by values of the variables, find an assignment of values to variables. Cryptarithmic problems (see Refs 3 and 28) are of this type, and so are problems of assigning interpretations (in terms of edge types, etc.) to features of visual scenes. Work on vision (qv) has resulted in procedures for solving constraint-satisfaction problems that are based on the notion of eliminating candidate solutions via a method of successive refinements of value sets of the variables (29,30). The elimination is achieved by reasoning with the constraints of the problem and by identifying subsets of the set of candidate solutions that are inconsistent with the constraints. The stronger the dependencies between the constraints, the more complex (and more exhaustive) the process of a priori elimination of solution candidates. Following the candidate-elimination stage, the problem solver can proceed with the process of finding a solution by searching in the reduced set of candidates using a generate-and-test schema or one of its variants. An analysis of processes for solving constraint-satisfaction problems shows that these processes involve simultaneous work in two spaces, i.e., in the space of solution structures (sets of assignments of values to variables) and the space of problem conditions (specified by the predicates and their relationships) (31). One of the important problems in this area is how to coordinate work in these two spaces, i.e., how to control the focus of attention between analysis of problem conditions that leads to restrictions on the space of solution candidates (and possibly to the identification of "promising" solution candidates) and generation of (parts of) solution candidates and testing them in light of problem conditions.

In DENDRAL the basic generate-and-test schema was modified and augmented to yield remarkably high performance in a task that is essentially of formation type. The main device used in this case was to constrain the generation of solution candidates a priori by suitable analysis of problem conditions. There are several other variants of generate and test that are extensively used in AI, each based on a different device for limiting the explosive generation of solution alternatives. An important such variant is the hill-climbing schema. In this schema feedback from the test step is used to guide the generator in its selection of the next solution-construction action. More specifically, the test step is used not only to establish whether a (completely specified) candidate solution satisfies the desired problem conditions but also to assess "how close" the candidate is to the goal of attaining the conditions. The generator focuses attention on the candidate with the strongest "closeness" assessment, and it performs solution-construction actions that amount to relatively small "modifications" of the candidate. In the subsequent cycle, attention focuses on the modified candidate with the strongest closeness assessment, and so on. The choice of a solution grammar (which characterizes the ways in which solutions can be modified) and of an assessment function have a strong impact



on the performance of hill-climbing procedures. Some of the difficulties encountered in these procedures are due to "local maxima," "plateaus," and "ridges" in assessment functions defined over the set of candidate solutions. These "topographical" features of assessment functions make it hard for a procedure to move toward a desired solution. Various techniques have been developed for alleviating (but not for eliminating) these difficulties.

**Optimization Problems.** A class of problems in which variants of hill-climbing procedures are frequently used is optimization problems, exemplified by the traveling-salesman problem. In these problems, in addition to various structural constraints imposed on the solution, there is also a minimality (or maximality) condition on a feature of the solution; for example, the desired tour in the traveling-salesman problem must be of minimum length. An interesting approach to these problems is that developed to obtain  $k$ -optimal tours for the traveling-salesman problem via a process of successive approximations (32). Here, processing takes place mainly in the space of solution structures, and "movements" in this space are in the form of local deformations of entire solutions: Up to  $k$  segments of a given candidate solution (a tour) are changed in a disciplined way so that no structural constraints are violated, and eventually a new candidate solution is reached that cannot be improved by any modification in up to  $k$  of its segments. Thus, the thrust of reasoning is from possible solution structures to problem conditions. Problem conditions have little *a priori* influence on the generation of solution candidates. Representations of solutions and their possible deformations and a disciplined handling of the process of successive approximations are of key importance.

**Shifts in Problem Formulation Viewed as Movement over Derivation-Formation Spectrum.** It is a common phenomenon in human problem solving that the formulation of a problem in a given domain changes as experience accumulates and as more knowledge about handling problems in the domain becomes available. Often, this can be seen as movement of the problem over the derivation-formation spectrum in the direction of the derivation end. This type of movement represents an important mode of shift in problem representation, which may result in a significant improvement in problem-solving performance. In general, growth in problem-solving expertise seems to involve the acquisition of more knowledge in a form that permits increased direct control of the solution-construction process by the problem conditions.

A good example of this type of representational shift can be seen in DENDRAL (25). Experience of work with DENDRAL has shown that as more knowledge becomes available in a form that is suitable to exert *a priori* control over the process of generating candidate solutions (via strengthening of the planning process), the system acquires additional selectivity and power. Furthermore, it became clear that the appropriate approach for strengthening the planning process was not through specification of a single general procedure for going from a broad data domain to a correspondingly very large space of hypotheses. What was needed was to partition the data domain into appropriate subdomains and to develop for each of the subdomains specialized data-analysis processes for constraining the set of hypotheses. Thus, expert, high-performance behavior was obtained in the present case via the development of several strong reasoning links from data to hypothe-

ses, each specialized to a subdomain of the problem. This type of approach has been extensively used in the design of expert systems that followed DENDRAL.

### Problems of Learning, Theory Formation, Analogy, and Abstraction

The issue of expertise acquisition, i.e., the transition from less expert, or even from relatively weak, to more expert or stronger problem-solving performance, is receiving increased attention in recent years. In particular, the problem of mechanizing certain aspects of expertise acquisition (or of strategy improvement) is being explored by several AI workers. One of the approaches studied is learning applicability conditions of problem-solving moves, that is, using problem-solving experience to redefine the applicability conditions of moves in a way that increases the selectivity of their application. For example, the LEX project is focusing on this problem in the context of a symbolic integration task (33). The key problem here is to learn a good definition for the domain of a move, i.e., to form a concept of the move's domain in terms of features of problem states. This is an instance of concept formation (see Concept learning) of the kind that received considerable attention in machine-learning research (see Learning, machine).

The area of machine learning has received "pulses of attention" by the research community since the early stages of work in AI. Recently, there is renewed interest in this area. Learning tasks have different forms depending on the nature of the body of information that constitutes input to the task, the language in which the output of learning is expressed and the rules and "biases" that control the learning process. Several researchers have pointed out that learning is an instance of problem solving (31,33). This is a fruitful conceptual view that permits ideas from other areas of problem solving to be transferred to learning tasks. For example, in a concept-formation problem the problem conditions have the form of given instances of the concept (positive and negative), and these instances must be consistent with the concept definition that the system is asked to find. Furthermore, the concept definition (i.e., the desired solution) must be constructed in a given language, and it must satisfy certain other global conditions, such as simplicity. Depending on how instances are used in the process of constructing a concept definition, the concept-formation problem has characteristics of a derivation or a formation problem. If the instances are used directly, together with rules of generalization, to come up with hypotheses about concept definitions, the problem is close to the derivation end of the derivation-formation spectrum. If the main thrust of reasoning can be seen as search in the space of concept definitions, with the instances used mainly for testing candidate concepts (hypotheses), the problem is close to the formation end of the spectrum. Concept-learning problems of the latter type are closely related to an important class of theory-formation problems, where a substantial part of the problem-solving effort involves search in the space of hypotheses/theories. What distinguishes the concept-learning problems from the theory-formation problems is that the latter use a language of hypotheses/theories that is generally more rich than the language used in the former.

Learning and theory formation are problem-solving processes that are needed to extract knowledge from problem-solving experience in a task domain so that problem-solving performance (expertise) in the domain can be strengthened. A

good understanding of properties of various problem-solving schemas—and in particular of “points of leverage” in these schemas where augmentation of knowledge can improve performance—is also needed in order to take advantage of the added knowledge. One aspect of theory formation that did not receive much attention as yet but that is clearly important for processes of strategy improvement (as well as for many areas of scientific work) is concept discovery. This activity can be regarded as selecting an “interesting” subdomain of phenomena for which it is deemed desirable to find a theoretical characterization/explanation. In general, theoretical studies of a “piece of the world” involve both the choice of a circumscribed domain of phenomena in the world that are to be expressed in some conceptual framework and also the finding of expressions within the framework for defining/explaining the domain of phenomena. The interplay between domain choice and the formulation of a theory for the domain is an interesting and complex process that is difficult to capture at present. A promising study in this area is Lenat’s work on AM, where the task is to discover “interesting” concepts in elementary number theory (34). The AM system starts with a small set of mathematical concepts and with rules for modifying definitions of existing concepts and for combining concepts. This defines a space of concepts, which is searched selectively via a generate-and-test schema. The generator is guided by a body of heuristic rules, and the test stage is guided by rules of “interestingness” that act as a filter for candidate concepts that are produced by the generator.

One of the important approaches to problem solving that is widely used involves finding a “similar problem”—for which a solution is known—and proceeding to find a solution to the original problem “in analogy to” the solution of the “similar problem.” This is a powerful approach that is receiving increased attention in AI (9). Much more work is needed in this area. A related approach is to solve a simplified version of the original problem and to use the solution of the simplified problem as a guide for solving the original. An attempt to mechanize this approach goes back to the early days of AI, where the logic theorist, whose task was to find proofs in elementary symbolic logic, used a planning stage that produced a skeletal proof, which then guided the construction of the final proof (35). In the early seventies the ABSTRIPS system was developed to facilitate the solution of complex planning problems by creating a simplified problem, where some of the preconditions of actions were stripped out, and by using its solution to find a complete solution for the original problem (36). This approach has conceptual similarities to the relaxed-reduction approach. In both approaches more work is needed, especially in identifying ways of moving from relaxed/simplified solutions to actual solutions.

### Concluding Comments

Problem solving is the art of using relevant knowledge in the attainment of desired goals. Within AI, work on problem solving focuses on techniques for solving exponentially hard problems in polynomial time by exploiting knowledge about the problem in a relatively small number of generic ways. Of key importance are mechanisms for acquiring, representing, and using knowledge. Systems are now being developed in which problem-solving performance in a domain will not be a stationary property of a system, but it will change (improve) with experience. Such a change will be mediated by learning and

theory-formation processes, which are themselves problem-solving processes. To mechanize and coordinate this variety of processes, it is essential to have a broad conceptual framework in which one can see clearly properties and relationships of various problem formulations, problem-solving methods, and procedures for constructing solutions.

### BIBLIOGRAPHY

1. S. Amarel, On the Representation of Problems and Goal-Directed Procedures for Computers, in R. Banerji and M. Mesarovic (eds.), *Theoretical Approaches to Non-Numerical Problem Solving*, Springer-Verlag, Heidelberg, pp. 179–244, 1970; also appears in *Commun. Am. Soc. Cybernet.* 1(2), 9–36 (July 1969).
2. S. Amarel, Problems of Representation in Heuristic Problem Solving; Related Issues in the Development of Expert Systems, in R. Groner, M. Groner, and M. W. Bischof (eds.), *Methods of Heuristics*, Lawrence Erlbaum, Hillsdale, NJ, pp. 245–349, 1983; also appears as Technical Report CBM-TR-118, LCSR, Rutgers University, New Brunswick, NJ, February 1981.
3. A. Newell and H. A. Simon, *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ, 1972.
4. J. McCarthy, The Inversion of Functions Defined by Turing Machines, in C. E. Shannon and J. McCarthy (eds.), *Automata Studies*, Annals of Mathematical Studies, Vol. 34, Princeton University Press, Princeton, NJ, pp. 177–181, 1956.
5. H. A. Simon, “The structure of ill structured problems,” *AI J.* 4, 181–201 (1973).
6. A. Tversky and D. Kahneman, “The framing of decisions and the psychology of choice,” *Science* 211, 453–458 (January 30, 1981).
7. G. Polya, *How To Solve It*, 2nd ed., Doubleday, Garden City, NY, 1957.
8. J. R. Hayes and H. A. Simon, Understanding Written Problem Instructions, in L. W. Gregg (ed.), *Knowledge and Cognition*, Lawrence Erlbaum, Potomac, MD, pp. 167–200, 1974.
9. J. G. Carbonell, Jr., A Computational Model of Analogical Problem Solving, in *Proceedings of the Seventh IJCAI*, Vancouver, B.C., pp. 147–152, 1981.
10. S. Amarel, An Approach to Heuristic Problem Solving and Theorem Proving in the Propositional Calculus, in J. Hart and S. Takasu (eds.), *Systems and Computer Science*, University of Toronto Press, Toronto, Canada, pp. 125–220, 1967.
11. S. Amarel, On Representations of Problems of Reasoning About Actions, in D. Michie (ed.), *Machine Intelligence*, Vol. 3, Edinburgh University Press, Edinburgh, U.K., pp. 131–171, 1968.
12. S. Amarel, Problem Solving and Decision Making by Computer: An Overview, in P. Garvin (ed.), *Cognition: A Multiple View*, Spartan, New York, pp. 279–329, 1970.
13. S. Amarel, Representations and Modeling in Problems of Program Formation, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 6, University of Edinburgh Press, Edinburgh, U.K., pp. 411–466, 1971.
14. E. Rich, *Artificial Intelligence*, McGraw-Hill, New York, 1983.
15. J. E. Laird and A. Newell, A Universal Weak Method, Technical Report No. 83-141, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, June 1983; also appears in summary form in *Proceedings of the Eighth IJCAI*, Karlsruhe, FRG, pp. 771–773, 1983.
16. A. Newell, Heuristic Programming: Ill Structured Problems, in J. Aronofsky (ed.), *Progress in Operations Research*, Vol. 3, Wiley, New York, pp. 360–414, 1969.
17. N. J. Nilsson, *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.
18. A. Newell, Reasoning, Problem Solving and Decision Processes:

- The Problem Space as Fundamental Category, in R. Nickerson (ed.), *Attention and Performance*, Vol. 8, Lawrence Erlbaum, Hillsdale, NJ, pp. 693–718, 1980.
19. N. J. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, 1980.
  20. J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, Reading, MA, 1984.
  21. M. Groner, R. Groner, and W. F. Bischof, *Approaches to Heuristics: A Historical Review*, in R. Groner, M. Groner, and W. F. Bischof (eds.), *Methods of Heuristics*, Lawrence Erlbaum, Hillsdale, NJ, pp. 1–18, 1983.
  22. G. W. Ernst and A. Newell, *GPS: A Case Study in Generality and Problem Solving*, Academic Press, New York, 1969.
  23. R. Korf, *Learning to Solve Problems by Searching for Macro-Operators*, Pitman, Marshfield, MA, 1985.
  24. E. D. Sacerdoti, *A Structure for Plans and Behavior*, Elsevier, New York, 1977.
  25. B. G. Buchanan and E. A. Feigenbaum, "DENDRAL and Meta-DENDRAL: Their applications dimension," *AI J.* (Special Issue on Applications to Science and Medicine) 11(1,2), 15–24 (August 1978).
  26. R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, *Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project*, McGraw Hill, New York, 1980.
  27. P. Szolovits (ed.), *Artificial Intelligence in Medicine*, Westview, Boulder, CO, 1982.
  28. H. A. Simon, *The Sciences of the Artificial*, MIT Press, Cambridge, MA, 1969.
  29. D. Waltz, Understanding Line Drawings of Scenes with Shadows, in *The Psychology of Computer Vision*, P. Winston (ed.), McGraw-Hill, New York, pp. 19–92, 1975.
  30. A. K. Mackworth, "Consistency of networks of relations," *AI J.* 8(1), 99–118 (1977).
  31. H. A. Simon and G. Lea, Problem Solving and Rule Induction: A Unified View, in L. W. Gregg (ed.), *Knowledge and Cognition*, Lawrence Erlbaum, Potomac, MD, pp. 105–127, 1974.
  32. S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," *Operat. Res.* 21(2), 498–516, (March–April 1973).
  33. T. M. Mitchell, "Learning and Problem Solving," *Proceedings of the Eighth IJCAI*, Karlsruhe, FRG, pp. 1139–1151, 1983.
  34. R. Davis and D. Lenat, AM: Discovery in Mathematics as Heuristic Search, *Knowledge-Based Systems in Artificial Intelligence*, McGraw-Hill, New York, Part 1, 1982.
  35. A. Newell, J. C. Shaw, and H. A. Simon, Empirical Exploration of the Logic Theory Machine: A Case Study in Heuristics, in E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, pp. 109–133, 1963.
  36. E. Sacerdoti, Planning in a Hierarchy of Abstraction Spaces, *Proceedings of the Third IJCAI*, Stanford, CA, pp. 412–422, August 1973.

S. AMAREL  
Rutgers University  
and DARPA/ISTO

## PROCESSING, BOTTOM-UP AND TOP-DOWN

"Bottom-up" vs. "top-down," "forward" vs. "backward," and "data-driven" vs. "goal-directed" are three pairs of modifiers for terms such as "chaining," "inference," "parsing," "processing," "reasoning," and "search." They express essentially the

same distinction, their difference lying in different metaphors drawn from different subareas of computer science and AI. The bottom-up–top-down distinction comes from parsing (qv); the forward–backward chaining distinction comes from rule-based systems (qv); goal-directed comes from problem solving (qv) and search (qv), and data-directed comes from discussions of control structures (qv). Now, however, they are virtually interchangeable.

One general way to consider the distinction is from within the paradigm of search. The basic issue of all search is to find a way to get from where you are to where you want to be. If you organize this by starting from where you are and search until you find yourself where you want to be, you are doing forward, data-directed, or bottom-up search. If you think about where you want to be, and plan how to get there by working backward to where you are now, you are doing backward, goal-directed, or top-down search. Notice that, having found the route during backward search, you still have to get to your goal. Although you are now moving in the forward direction, this is not forward search because all the search was already done in the backward direction.

Another general way to consider the distinction is from within the paradigm of rule-based systems (qv). A generic rule can be thought of as having a set of antecedents and a set of consequents. When the rule-based system notices that all the antecedents of a rule are satisfied, the rule is triggered and may fire (whether all triggered rules actually fire depends on the specifics of the rule-based system). When the rule fires, the consequent propositions are added to the knowledge base and the consequent actions are performed. These steps of triggering and firing happen as just described regardless of whether the rule-based system is using forward (or data-directed or bottom-up) reasoning or backward (or goal-directed or top-down) reasoning. To make the distinction, it is useful to isolate the step of rule activation. Only activated rules are subject to being triggered. In forward (or data-directed, or bottom-up) reasoning, whenever new data is added to the system, the data is matched against all antecedents of all rules (actual systems are more efficient than this sounds). If the data matches an antecedent of a rule, that rule is activated (if it is not already activated), and if all antecedents of the rule are now satisfied, the rule triggers. When a rule fires, the consequent propositions that are added to the knowledge base are treated like new data, matched against antecedents, and may cause additional rules to be activated and to be triggered. In backward (or goal-directed, or top-down) reasoning, rules are not activated when data is added. Rather, when a query is asked of the system, or the system is asked to do something, the query (or goal) is matched against all consequents of all rules (again, actual systems are more efficient than this sounds). If the query matches a consequent of a rule, the rule is activated, all its antecedents are treated as new queries or goals (now called subqueries, or subgoals), and may activate additional rules. Whenever a query or subquery matches an unconditional proposition in the knowledge base, it is answered, and if it came from an antecedent, the antecedent is now known to be satisfied. As soon as all antecedents of some rule are known to be satisfied, the rule triggers and may fire. When a rule fires, the queries that activated it are answered, and now other antecedents may be known to be satisfied, and their rules might be triggered. Notice that the triggering and firing of a rule always seems to happen in a "forward" direction, due to the significance of "antecedents" vs. "consequents," but what dis-

tinguishes forward from backward chaining is when the rule is activated.

Some of the history of these terms is discussed below, followed by explanations of the distinctions via examples of parsing, rule-based systems, and search. Then there will be some comparative comments and, finally, some discussion of mixed strategies.

### History of Terms

**Bottom-Up versus Top-Down.** The earliest published occurrence of the phrase "top-down vs. bottom-up" seems to have been in a 1964 paper by Cheatham and Sattley (1), although the term "top-down," at least, seems to already have been in use:

*The Analyzer described in this paper is of the sort known as "top down," the appellation referring to the order in which the Analyzer sets its goals. . . . The order in which a "bottom up" Analyzer sets its goals is much more difficult to describe. . . .* (2)

It took a while, however, for these terms to become fully accepted. In 1965 Griffiths and Petrick (3) used the terms "bottom-to-top" and "top-to-bottom":

*There are many ways by which the typology of recognition algorithms for [Context Free] grammars can be approached. For example, one means of classification is related to the general directions in which creation of a structural description tree proceeds: top-to-bottom, bottom-to-top, left-to-right and right-to-left.*

However, in 1968, they use the terms "bottom-up" and "top-down" in a title (4).

Also in 1968, Knuth (5) used the term "bottom-up" to describe a function "which should be evaluated at the sons of a node before it is evaluated at the node" and "top-down" to describe a function  $f$  as "one in which the value of  $f$  at node  $x$  depends only on  $x$  and the value of  $f$  at the father of  $x$ ." Knuth encloses these terms in quotes.

By 1970 Early (6) used the phrase "the familiar top-down algorithm" without attribution in the abstract of an article, and by 1973 "The Top-down Parse" and "The Bottom-up Parse" appear as section titles in a text (7).

Bottom-up and top-down parsing are referred to as the "morsel" and the "target" strategies, respectively, in Ref. 8, p. 87, where the Predictive Analyzer of Kuno and Oettinger (9) is cited as an early example of the target strategy, and "the algorithm due to John Cocke and used in Robinson's PARSE program (10) was the earliest published example of a program using a morsel strategy." Calingaert (11) cites Lucas (12) as the first described use of recursive descent parsing, which is a form of top-down parsing.

**Forward versus Backward Chaining.** The terms "forward chaining" and "backward chaining" almost surely come from Newell, Shaw, and Simon's Logic Theory Machine (LT) paper, first published in 1957 (13). They discuss four methods used by LT to help find a proof of a formula in propositional calculus (qv). The last two methods discussed are called "the chaining methods":

*These methods use the transitivity of the relation of implication to create a new subproblem which, if solved, will provide a proof*

*for the problem expression. Thus, if the problem expression is "a implies c," the method of forward chaining searches for an axiom or theorem of the form "a implies b." If one is found, "b implies c" is set up as a new subproblem. Chaining backward works analogously: it seeks a theorem of the form "b implies c," and if one is found, "a implies b" is set up as a new subproblem (14).*

This is not exactly the characterization of forward and backward chaining given above, but the essential idea is there. In both methods if a certain theorem is found, an appropriate subproblem is set up. In forward chaining the theorem is found by matching its antecedent, and its consequent is involved in the new subproblem, whereas in backward chaining the theorem is found by matching its consequent, and its antecedent is involved in the new subproblem. Finding a theorem is analogous to activating a rule in the characterization described at the beginning of this entry.

The rule-based system version of forward and backward chaining grew out of the LT version via the production system architecture of problem-solving systems promulgated in Ref. 15.

**Goal-Directed Processing.** The notion of goal-directed behavior surely comes from psychology. In a 1958 psychology text the following description of problem solving occurs:

*We may have a choice between starting with where we wish to end, or starting with where we are at the moment. In the first instance we start by analyzing the goal. We ask, "Suppose we did achieve the goal, how would things be different—what subproblems would we have solved, etc.?" This in turn would determine the sequence of problems, and we would work back to the beginning. In the second instance we start by analyzing the present situation, see the implications of the given conditions and lay-out, and attack the various subproblems in a "forward direction" (16).*

Also, goals and subgoals are discussed in the section on motivation:

*The person perceives in his surroundings goals capable of removing his needs and fulfilling his desires. . . . And there is the important phenomenon of emergence of subgoals. The pathways to goals are often perceived as organized into a number of subparts, each of which constitutes an intermediate subgoal to be attained on the way to the ultimate goal. (17)*

Cheatham and Sattley, who are mentioned above as publishing probably the first use of "bottom up vs. top down," also compared top-down parsing to goal-directed behavior:

*In our opinion, the fundamental idea—perhaps "germinal" would be a better word—which makes syntax-directed analysis by computer possible is that of goals: a Syntactic Type is construed as a goal for the Analyzer to achieve, and the Definiens of a Defined Type is construed as a recipe for achieving the goal of the Type it defines. . . . Needless to say, this use of the term "goal" is not to be confused with the "goal-seeking behavior" of "artificial intelligence" programs or "self-organizing systems" (18).*

Needless to say, the several uses of "goal" indeed have much in common.

**Data-Driven Processing.** The term “data-driven” seems to have been introduced by Bobrow and Norman in a paper on the processing of memory schemata:

*Consider the human information processing system. Sensory data arrive through the sense organs to be processed. Low-level computational structures perform the first stages of analysis and then the results are passed to other processing structures. . . . The processing system can be driven either conceptually or by events. Conceptually driven processing tends to be top-down, driven by motives and goals, and fitting input to expectations; event driven processing tends to be bottom-up, finding structures in which to embed the input (19).*

They go on to use “conceptually driven” and “top-down” as synonymous and “event-driven,” and “data-driven” interchangeably and synonymously with “bottom-up.”

### Examples

**Parsing.** The simple grammar of Figure 1 and the sentence “They are flying planes” will be used to illustrate and explain the difference between top-down and bottom-up parsing. The grammar is presented as a set of rules, numbered for the purposes of this discussion. The direction of the arrows is traditionally shown as if the grammar were being used for generation of sentences. For parsing, the rules are backward—the antecedents on the right side and the consequent on the left side, as in PROLOG (see Logic programming). For example, rule 1 can be read “If a string consists of a noun phrase (NP) followed by a verb phrase (VP), then the string is a sentence (S).”

Top-down parsing begins with S, the initial symbol, which will be the root of the parse tree. This is equivalent to establishing the goal of finding that the string of words is a sentence. Rule 1 says that every sentence will consist of a noun phrase (NP) followed by a verb phrase (VP). Whenever there is a choice, the lowest numbered rule is tried first, and the rule is expanded left to right. Therefore, the next subgoal generated is that of finding an initial string of the sentence that is a NP. Rule 2 is activated, followed by rule 8. This situation is shown in Figure 2a. Since “planes” does not match “they,” the algorithm backs up, and in place of rule 2, rule 3 is activated followed by rule 9, which succeeds. Next, the algorithm returns to rule 1 and generates the subgoal of finding a VP. Rules 5, 6, and 12 are activated, and rule 12 succeeds. This stage is shown in Figure 2b. The rest of the top-down parse is shown in stages in the rest of Figure 2.

Bottom-up parsing begins with the words in the sentence.

1.  $S \rightarrow NP VP$
2.  $NP \rightarrow N$
3.  $NP \rightarrow PRO$
4.  $NP \rightarrow ADJ N$
5.  $VP \rightarrow VT NP$
6.  $VT \rightarrow V$
7.  $VT \rightarrow AUX V$
8.  $N \rightarrow planes$
9.  $PRO \rightarrow they$
10.  $ADJ \rightarrow flying$
11.  $AUX \rightarrow are$
12.  $V \rightarrow are$
13.  $V \rightarrow flying$

Figure 1. An example grammar.

Again, additional ordering decisions must be made, so the left-most possibility is tried first, as is the lowest numbered rule when there is a choice. So, the first thing that happens is that the first word of the sentence, “they,” matches the antecedent of rule 9, which fires, analyzing “they” as a pronoun (PRO). Then rule 3 fires, analyzing “they” as a NP. NP matches antecedents in rules 1 and 5, but neither of these is triggered yet, and the parse moves on to “are.” This causes rule 11 to fire. (Although rule 12 is also triggered, it does not fire due to the ordering rules.) Then rule 10 fires, followed by rules 8 and 2. This stage is shown in Figure 3a. At this point, nothing more can be done with the string NP+AUX+ADJ+NP, so there is backtracking (qv) to the most recently triggered but unfired rule, which is rule 4. Nothing can be done with the string NP+AUX+NP, so again the most recently triggered unfired rule fires, which now is rule 13, which analyzes “flying” as a V. Then rule 6 fires, followed by rule 5. This is shown in Figure 3c. Nothing can be done with the string NP+AUX+VP, so rule 7 fires, analyzing “are flying” as a single VT. Then rule 5 fires again, followed by rule 1, and the parse, shown in Figure 3d, is complete.

The purpose of this example was to compare top-down with bottom-up parsing. The use of left-to-right order, numerical order of the rules, and chronological backtracking was merely to keep the two algorithms as similar as possible although causing them to result in different parses.

**Rule-Based Systems.** Forward vs. backward chaining and data-directed vs. goal-directed processing will be compared using some made-up rules for how to spend the evening. These are shown in Figure 4 in the usual rule order, with the antecedents to the left of the arrow and the consequent to the right. For example, rule 1 says that if there is a good movie on TV and I have no early appointment the next morning, then I enter “Late-Movie-Mode.”

For the examples of forward and backward chaining, it is assumed that all triggered rules fire and that processing is in parallel.

Suppose a forward-chaining system with these rules is first told that I have no early appointment. Rules 1 and 2 are activated. Suppose the system is then told that I need to work. Rule 3 is activated, and rule 2 is triggered and fired, concluding that I am in Late-Work-Mode. This activates, triggers, and fires rules 5 and 6, concluding that I should return to the office and stay up late.

To do the same problem using backward chaining, suppose that the system was first told that I had no early appointment and had work to do and then was asked whether I should return to the office. This query would activate rules 6 and 7, which would generate the subgoals Late-Work-Mode? and Work-at-Office-Mode? These would activate rules 2 and 3, generating the subgoals No-Early-Appointment?, Need-to-Work?, and Need-References? The first two would be satisfied, resulting in rule 2 triggering and firing. This would cause Late-Work-Mode to be satisfied, triggering and firing rule 6, and concluding that I should return to the office.

Notice that in forward inference more conclusions are generated, whereas in backward inference more subgoals are generated. Since the data were the same for the two examples, the same rules were fired, but different rules were activated.

**Search.** Forward and backward search can be illustrated with a water-jug problem. For this problem there are two jugs, one capable of holding 3 gallons and one capable of holding 4

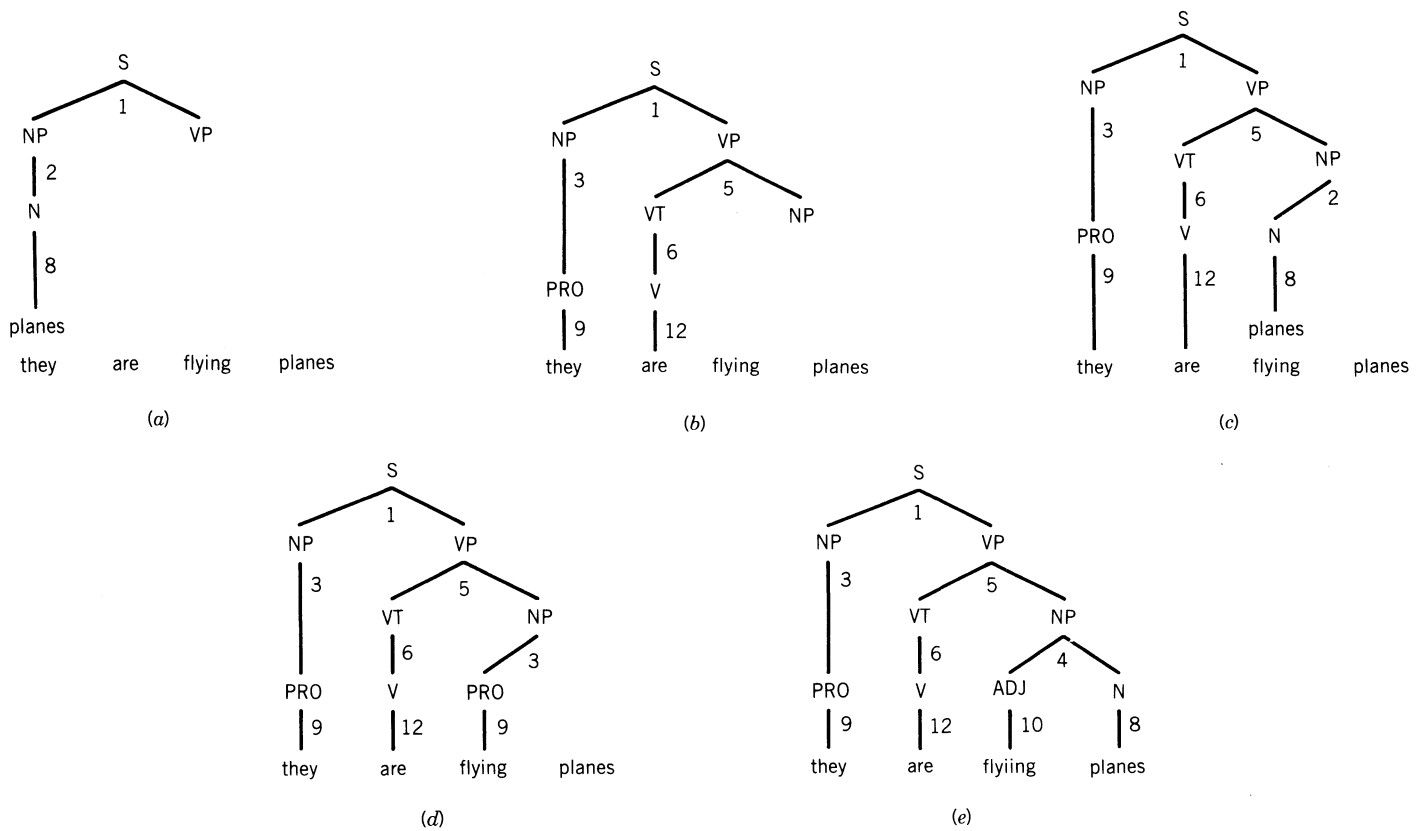


Figure 2. Top-down parsing.

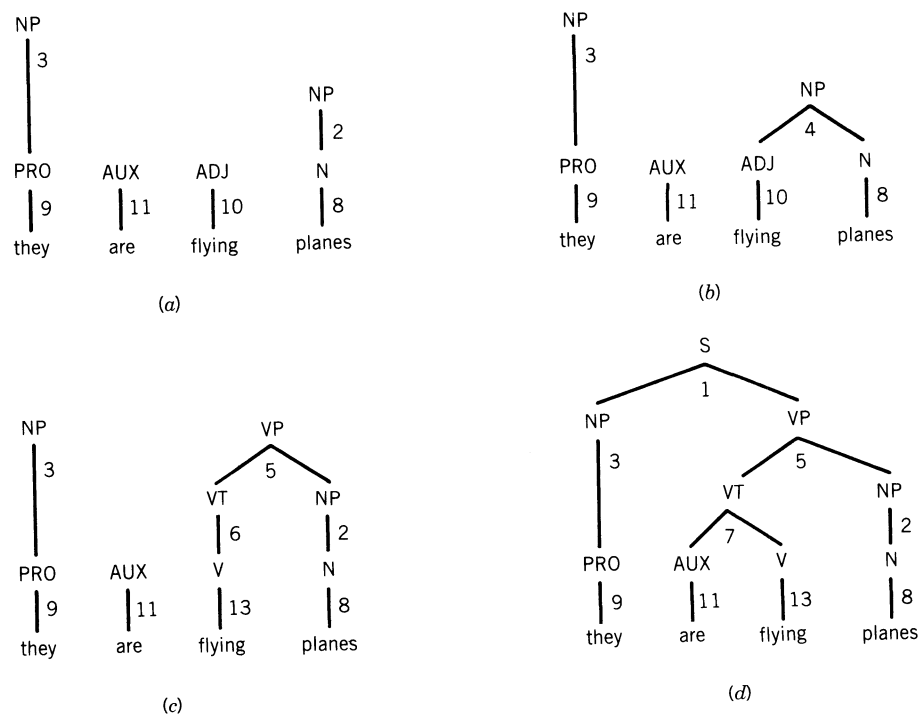


Figure 3. Bottom-up parsing.



1. Good-Movie-on-TV & No-Early-Appointment  $\rightarrow$  Late-Movie-Mode
2. No-Early-Appointment & Need-to-Work  $\rightarrow$  Late-Work-Mode
3. Need-to-Work & Need-References  $\rightarrow$  Work-at-Office-Mode
4. Late-Movie-Mode  $\rightarrow$  Stay-Up-Late
5. Late-Work-Mode  $\rightarrow$  Stay-Up-Late
6. Late-Work-Mode  $\rightarrow$  Return-to-Office
7. Work-at-Office-Mode  $\rightarrow$  Return-to-Office

Figure 4. Rules for the evening.

gallons. The legal operations are filling the 3-gallon jug from a water tap (symbolized as f3); filling the 4-gallon jug from the water tap (f4); emptying the 3-gallon jug by pouring out all its contents (e3); emptying the 4-gallon jug by pouring out all its contents (e4); pouring the contents of the 3-gallon jug into the 4-gallon jug until either the 3-gallon jug is empty or the 4-gallon jug is full, whichever happens first (p34); or pouring the contents of the 4-gallon jug into the 3-gallon jug until either the 4-gallon jug is empty or the 3-gallon jug is full, whichever happens first (p43). Each state of the problem is represented by a pair of integers showing the contents of the 3-gallon jug and then the contents of the 4-gallon jug. For example, (1, 4) represents the state in which there is 1 gallon in the 3-gallon jug and 4 gallons in the 4-gallon jug. If operator p43 is applied to this state, the resulting state is (3, 2). The particular problem under consideration is that of getting from state (0, 0) to state (0, 2).

Figure 5 shows the state-space representation of this problem, assuming a parallel breadth-first search that stops as soon as the goal is found. No operator is shown that would move from a state to a state at the same or an earlier level of the search tree. For example, from state (3, 1) operator f4 would go to state (3, 4), but that is on the same level as (3, 1), and operator e4 would go to state (3, 0), but that is on an earlier level. What level a state is on, of course, depends on where the search started.

Figure 5a shows a forward search from (0, 0) until (0, 2) is found. Figure 5b shows a backward search from (0, 2) to (0, 0). Notice that, in this example, the same states are explored, but in a slightly different order. Notice also the difference between searching backward to find a way of getting from (0, 0) to (0, 2) and searching forward to find a way of getting from (0, 2) to (0, 0). In the latter case, one operator, namely e4, would suffice.

### Comparisons

**Efficiency.** Whether bottom-up (or forward, or data-driven) processing is more efficient than top-down (or backward, or goal-directed) processing depends on the way the search space branches. If the average state has more successors than predecessors, backward search will be more efficient. If it has more predecessors than successors, forward search will be more efficient. To consider an extreme, if the search space forms a tree

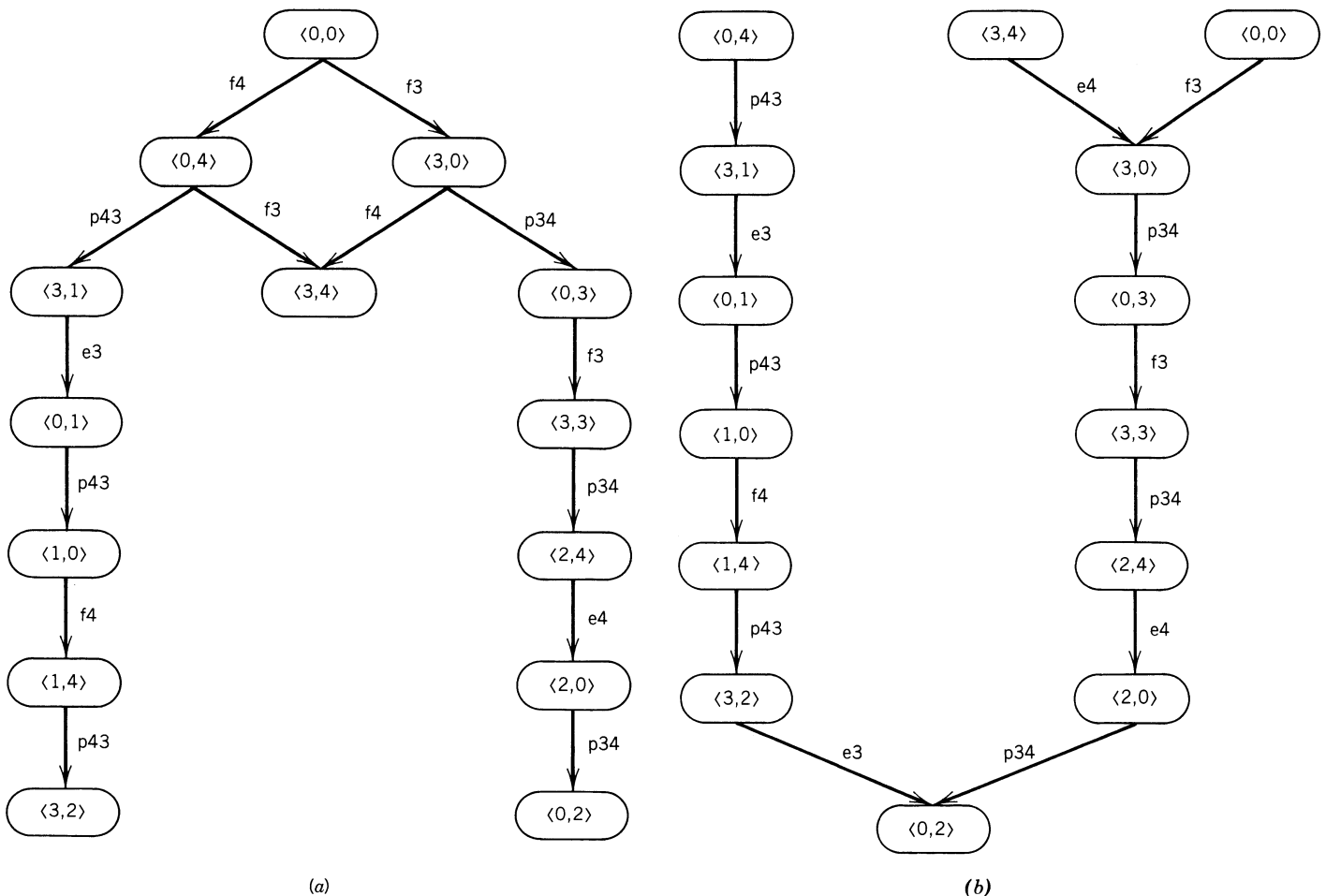


Figure 5. (a) Forward search. (b) Backward search.

rooted in the start state, a forward search will have to search a large part of the tree, whereas a backward search will only have to search up a linear branch.

**Pattern Matching and Unification.** In rule-based systems or reasoning systems, the choice of forward vs. backward chaining affects the difficulty of the required pattern-matching (qv) routines. In forward chaining, or data-driven reasoning, one is always asserting new facts to the system, and these have no free variables. Similarly, when rules fire, the newly inferred facts have no free variables. Therefore, one is always matching antecedents that may have variables against facts with no variables. Pattern matching two symbol structures when only one might have variables is a fairly simple routine.

On the other hand, in backward-chaining systems one often asks "wh" questions, such as "What shall I do this evening?" or "What organism is infecting this patient?" If the rules are represented in predicate logic (qv) rather than in propositional logic (qv), this involves matching a question with a variable against consequents with variables. Subgoals may also have variables, so in general, back-chaining systems must be written to match two symbol structures, both of which may have variables, and this requires the unification algorithm, which is considerably more involved than simple pattern matching.

### Mixed Strategies

**Bidirectional Search.** If it is not clear whether forward or backward search would be better for a particular application, bidirectional search (qv) is probably appropriate. Essentially, bidirectional search starts from both the start state and the goal state and searches from both ends toward the middle.

"Predictive syntactic analysis" is described in Ref. 20, where the first computer implementation is ascribed to Ref. 21. The bidirectional nature of this technique may be inferred from Bobrow's description that "an initial prediction is made that the string to be scanned is a sentence. From this prediction and the initial word in the sentence . . . more detailed predictions are made of the expected sentence structure" (22).

**Bidirectional Inference.** Another kind of bidirectional processing uses the initial data to activate rules that then trigger backward chaining through their other antecedents (23). Subgoals that match neither consequents nor data can remain as demons (qv) to be satisfied by new, later data. The system can be designed so that data that satisfy demons (antecedents of activated rules) do not activate additional inactive rules, thus focusing future forward inference on rules that take previous context into account.

**Left-Corner and Expectation-Based Parsing.** When the bidirectional style of inference is applied to parsing, one gets what is called left-corner parsing. In terms of the parsing example of the section Parsing, the system would first look at "they," find rule 9 as the only rule to account for it, then find rule 3 to be the only way to account for a PRO, and then find rule 1 as the only rule with a consequent side that starts with a NP. Next the system would try to parse "are flying planes" as a VP top-down. This is also very similar to expectation-driven parsing (qv) since the rules activated top-down form expectations of what will be in the rest of the sentence based on what actually occurred earlier in the sentence.

### Conclusions

The control structure of an AI system that does reasoning, parsing, problem solving, or search is often organized into one of two basic approaches. One approach is called bottom-up, forward, or data-driven. The other is called top-down, backward, or goal-directed. The distinction is most easily understood as whether search is from goal to start or if rules are activated by their consequents or their antecedents.

Issues of efficiency or ease of implementation may decide which approach to take in a particular application, but mixed strategies are also possible.

### BIBLIOGRAPHY

1. T. E. Cheatham and K. Sattley, Syntax-Directed Compiling, *Proceedings of the Spring Joint Computer Conference Washington, DC*, Spartan Books, Baltimore, MD, pp. 31-57, 1964.
2. Reference 1, p. 55.
3. T. V. Griffiths and S. R. Petrick, "On the relative efficiencies of context-free grammar recognizers," *CACM* 8(5), 289-300 (May 1965).
4. T. V. Griffiths and S. R. Petrick, Top-Down Versus Bottom-Up Analysis, in A. J. H. Morrell (ed.), *Information Processing 68: Proceedings of IFIP Congress 1968*, North-Holland, Amsterdam, pp. 437-442, 1969.
5. D. E. Knuth, *The Art of Computer Programming*, Vol. 1, *Fundamental Algorithms*. Addison-Wesley, Reading, MA, p. 362, 1968.
6. J. Early, "An efficient context-free parsing algorithm," *CACM* 13(2), 94-102 (February 1970).
7. F. W. Weingarten, *Translation of Computer Languages*, Holden-Day, San Francisco, 1973.
8. K. Sparck Jones and M. Kay, *Linguistics and Information Science*, Academic Press, New York, 1973.
9. S. Kuno and A. G. Oettinger, Multiple-Path Syntactic Analyzer, in C. M. Popplewell (ed.), *Information Processing-1962*, North-Holland, Amsterdam, pp. 306-312, 1963.
10. J. J. Robinson and S. Marks, PARSE: A System for Automatic Analysis of English Text, RM-4564-PR, Rand Corporation, Santa Monica, CA, 1965.
11. P. Calingaert, *Assemblers, Compilers, and Program Translation*, Computer Science Press, Rockville, MD, 1979.
12. P. Lucas, "Die Strukturanalyse von Formeluebersetzern," *Elektron. Rechenanl.* 3, 159-167 (1961).
13. A. Newell, J. C. Shaw, and H. A. Simon, Empirical Explorations with the Logic Theory Machine: A Case Study in Heuristics, *Proceedings of the Western Joint Computer Conference*, Los Angeles, CA, pp. 218-239, 1957. Reprinted in E. A. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, pp. 109-133, 1963.
14. Reference 13, pp. 117-118.
15. A. Newell and H. A. Simon, *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ, 1972.
16. D. Krech and R. S. Crutchfield, *Elements of Psychology*, Knopf, New York, p. 383, 1958.
17. Reference 16, pp. 218-219.
18. Reference 1, p. 33.
19. D. G. Bobrow and D. A. Norman, Some Principles of Memory Schemata, in D. G. Bobrow and A. Collins (eds.), *Representation and Understanding: Studies in Cognitive Science*, Academic Press, New York, pp. 138-140, 1975.
20. D. G. Bobrow, Syntactic Theories in Computer Implementations,

- in H. Borko (ed.), *Automated Language Processing*, Wiley, New York, pp. 215–251, 1967.
21. I. Rhodes, "A new approach to the mechanical syntactic analysis of Russian," *Mechan. Transl.* **6**, 33–50 (1961).
  22. Reference 20, p. 235.
  23. S. C. Shapiro, J. Martins, and D. McKay, Bidirectional Inference, *Proceedings of the Fourth Annual Conference of the Cognitive Science Society*, Ann Arbor, MI, pp. 90–93, 1982.

S. C. SHAPIRO  
SUNY at Buffalo

**PRODUCTION SYSTEMS.** See Expert systems; Programming environments; OPS-5.

**PROGRAM MODIFICATION.** See automatic programming.

## PROGRAMMING ASSISTANTS

A programming assistant is more than a conventional programming environment (qv) and less than an automatic programming (qv) system. A programming environment, consisting of an editor, compiler, debugger, and perhaps static analyzer, provides less substantial support than that provided by a programming assistant. An automatic programming system, by removing the need for programming, is more than merely an assistant to a human programmer.

A programming assistant may provide several types of support. One large class includes clerical and bookkeeping support. For example, a syntax-oriented editor helps the programmer avoid syntax errors (e.g., Ref. 1). INTERLISP's Programmer's Assistant includes a mechanism for storing, recalling, redoing, and undoing previous commands (2,3). Such systems clearly provide substantial support. However, their implementation does not rely on AI techniques, and they therefore will be excluded from the scope of this entry. Thus, for the purposes of this discussion, the following definition is used:

*A programming assistant is a computer system that provides substantial support to a skilled programmer in the nonclerical aspects of programming.*

In order to provide such nonclerical support, the assistant must have access to a substantial amount of knowledge about programming, the application domain, or both. Thus, the focus here is on knowledge-based programming assistants. Since the boundary between programming assistants and automatic programming is hard to draw precisely, the interested reader should consult the entry on automatic programming (see Automatic programming).

Given this definition, there are no knowledge-based programming assistants in routine use today, although many experimental systems have been built. Therefore, this entry focuses on the state of the art in research on programming assistants. The state of the art is reviewed along two dimensions: the programming tasks supported and the kinds of knowledge embodied in the systems. Several experimental systems are discussed in more detail, and the prospects for the future are considered.

## Programming Tasks

Programming tasks are often classified into two major categories. Programming in the small is concerned with the development by an individual programmer of relatively small programs (a few thousand lines) within a short period of time (a few months). Programming in the large is concerned with the development by groups of people of large software systems with a long intended lifetime. The major programming tasks differ for the two categories.

**Programming in the Small.** In programming in the small the major programming tasks and the type of support that would be expected from a knowledge-based programming assistant are described below.

**Specification.** What exactly should the program do? How should exceptional cases be handled? When a formal specification technique is used, there are two types of support that would be useful. One involves expressing the specification formally: Formal specifications seem important from a machine's point of view but are often difficult for people to work with; supporting informal expressions of more formal specifications would be quite valuable. One example of this is  $\phi_0$ , which allows the user to express certain mathematical relationships in a domain-specific form that is natural to the user (4). The other type of support involves testing the adequacy and consistency of the specification. Examples of such systems are the Gist specification paraphraser, which produces a natural-language paraphrase of a formal specification (5), and the Gist inference engine and symbolic evaluator, which draw conclusions about a formal specification and attempt to recognize those that would be interesting or surprising (6,7).

**Implementation.** What algorithms and data structures should be used? How can they be expressed in the syntax of the target language? For this task the primary type of support involves detailed knowledge of programming techniques. Such knowledge has been codified in the context of work on programming assistants (e.g., Refs. 8 and 9), program-transformation systems (see Ref. 10), and automatic programming (e.g., Refs. 11–14). Knowledge of the syntax of the target language is, of course, also required, but this is primarily of a clerical nature.

**Testing.** Does the implementation perform the specified computation? Is it robust enough for routine use? This activity depends on the existence of some sort of specification. Given a formal specification, it may be possible to verify that a program meets the specifications. Such techniques have been used to verify a few programs of substantial size but generally rely on human guidance and require significant computational support, primarily in the form of theorem provers (e.g., Ref. 15) (see Theorem proving). If there is no formal specification, the best that can be done is to be thorough in the selection and execution of test cases. One form of potential support would be a domain-specific example generator, but to date, no programming assistants provide such support.

**Optimization.** How efficient is the implementation? Can its efficiency be improved? Formal efficiency analysis is possible for certain types of programs. Examples of efficiency analyzers include the LIBRA component of the PSI system (16) and Paige's work on supercompilers (17). Improving a program's efficiency requires knowledge of alternative implementation techniques, and some of the systems cited earlier include

knowledge of the performance characteristics of alternative programming techniques (e.g., Ref. 9).

**Programming in the Large.** A programming-in-the-large project generally includes numerous programming-in-the-small subprojects, and hence includes the same tasks. In addition, however, there are several tasks oriented toward the increased need for communication and coordination:

**Requirements Analysis.** What are the key concepts and relationships of the domain? What are the users' needs? For this activity the most important type of support is assistance in expressing and formalizing knowledge of the domain. Fundamentally, this is no different from representing domain knowledge (qv) for other purposes (e.g., to build an expert system (qv)), and there has been very little work in this area aimed directly at providing support for programmers, except for preliminary theoretical work (e.g., Refs. 18 and 19).

**Design.** How can the problem be decomposed into manageable pieces? How do the pieces interact? The most important support for the design task would involve recording the design decisions and the motivations for them. Although there is widespread recognition of the importance of this issue (20), there are relatively few concrete results with broad applicability. Of the work that has been done, some has been oriented specifically toward programming (e.g., Refs. 21-23), and some has been in the context of other design tasks (see also Computer-aided, design).

**Integration.** Do the pieces interact in the intended way? What should be done when they do not? The best form of support for this task would involve analysis tools that could help track down the causes of undesirable interactions. To date, no substantial work has been done in this area.

**Maintenance.** Why does the program not function as intended? How can it be fixed so that it does? Maintenance is one of the most time-consuming tasks in software engineering, both because of the difficulty of diagnosing problems and because of the difficulty of predicting the effects of changes in the code. The single most important type of support that could be provided would be in the form of a repository of information about the various design and implementation decisions and the motivations for them, essentially the same sort of support required during the design task itself.

**Evolution.** Even though the program matches the specifications, does the program really satisfy the requirements? What should be done when the requirements change? Evolution is the other major time-consuming task in software engineering. The issues in evolution are similar to those of maintenance except that they are more complex since they involve a considerable amount of domain knowledge. Thus, the most important form of support would be assistance in managing a domain-specific knowledge base, essentially the same sort of support needed for requirements analysis.

**Project Management.** How can the amount of programming effort be predicted? How should the activities of a large group of people be coordinated? These issues are essentially the same as those involved in office automation (qv), and little work has been done in this area that is specifically related to programming, except for some preliminary experimental work (24).

**Types of Support.** In summarizing the types of support required for the various tasks, it can be seen that they fall into three general categories:

Several different types of knowledge must be represented explicitly and made available to the programming assistant. These are discussed in more detail under Areas of Knowledge, below.

Several different types of program manipulators are helpful, including algebraic manipulators, efficiency analyzers, symbolic executors, and verification systems. Each of these involves the development and use of specific techniques and algorithms, and in some cases practical systems have been built.

A mechanism for recording and retrieving information about the design and implementation decisions that have been made during the development of a specific piece of software. Such a mechanism would be quite complicated, both because of the sheer volume of information and because of the need for content-based retrieval. To date, unfortunately, relatively little has been done except for a few preliminary studies.

In addition, there is a need for integrating the various support facilities into a coherent whole. Building such an integrated system requires a coherent model of the overall software development process, a topic about which there is little agreement in the software-engineering community.

### Areas of Knowledge

In developing a knowledge-based system, it is often useful to separate different types of knowledge, both to focus the development effort and to increase the modularity and reusability of the final system. In the case of knowledge-based programming assistants, there are four primary areas of knowledge:

**Programming.** Obviously, programming requires knowledge about programming techniques. This includes techniques for implementing abstract data structures and algorithms, techniques for designing complex systems, and techniques for analyzing the efficiency of an implementation. To date, most work in this area has focused on the use of refinement or transformation rules (e.g., Refs. 21 and 22). **Target Language.** In addition, programming requires knowledge of the language in which the program is to be written. This includes knowledge about the syntax of the language and about the relative costs of different constructs in the language.

**Mathematics.** At various steps in the programming process, knowledge of mathematics plays a crucial role. For example, it is needed in order to develop formal specifications. In addition, it is fundamental to many types of analysis techniques, such as efficiency analysis. In many cases the mathematical techniques are well understood.

**The Domain.** The role of domain knowledge (qv) is more subtle than that of the other areas but very pervasive nonetheless. The tasks of requirements analysis and evolution are primarily concerned with domain knowledge. During design and implementation, domain knowledge is needed to make suitable choices between alternative techniques. To date, only some preliminary theoretical work has been done in this area (18,25).

In summary, some techniques for representing and using programming knowledge are relatively close to being of practical

value, requiring primarily refinement and testing. In contrast, techniques for the representation and use of domain knowledge require substantial basic research before they can be made available to a programming assistant.

### Experimental Programming Assistants

Although the major issues seem to be identified and to some degree understood, experience with functioning programming assistants has been limited to a relatively small number of systems developed and used only in experimental settings. In this section five of these will be described briefly.

**Programmer's Apprentice (PA).** The Programmer's Apprentice was designed to support experienced human programmers by analyzing code written by the programmer (8). At the heart of the system is a representation, called a plan, for describing programs and the relationships among their parts. For example, the plan for a program to delete an element from a bucket hash table has three parts: the first hashes on the key to find the index of the bucket; the second retrieves the bucket corresponding to the index; and the third removes the element from the bucket. In addition to control and data flow, a plan includes information about the purposes of different parts. In the hash-table example the first part is a prerequisite for the second, which is a prerequisite for the third; the third part achieves the goal of the entire program. The Programmer's Apprentice also includes a surface analyzer that translates code in a traditional language into plans and a large library of plans for a variety of programming techniques (26). In a subsequent experiment the library was used to support programmers during the initial coding process by making the knowledge available for synthesis as well as analysis (27).

**Data-Representation Advisor (DRA).** Katz and Zimmerman have developed a data-representation advisor (9). The system has three components: a set of abstract data types and possible operations on them, a set of base representations and efficient operations on them, and an algorithm for determining an appropriate combination of base cases for a given specification. For example, suppose the user wishes to represent an aggregate data structure (called S) that supports the following operations:

```
InsertByIndex(S,X)
FetchIndex(S,X)
DeleteMinimum(S)
Size(S)
```

The advisor selects a set of candidate abstract data types that cover the operations (vector, min-priority queue, counter). The advisor then asks specific questions to determine the best concrete representations for the abstract data types (binary tree, heap) and to determine whether certain simplifications are possible (eliminate counter). Finally, the advisor adds pointers from elements in one representation to elements in the other, thereby designing a complex linked structure that allows efficient execution of all operations.

**Glitter.** Glitter is a semiautomated assistant for a programmer using the transformational approach to implementation

(28). It was motivated by experience with a manually guided system for transforming Gist specifications into efficient code (22). Since many transformations were often required for what seem to be straightforward programming techniques, a considerable burden was placed on the programmer who was guiding the process. Glitter is intended to automate a significant fraction of this effort. Glitter's knowledge of implementation techniques is encoded in terms of methods and rules for selecting among them. For example, a relation defined on a set of objects could be implemented using either of two methods: constant updating of a database or computation of the relation whenever needed. Corresponding to such methods are selection rules. For example, if the relation is very large (i.e., many pairs of objects satisfy the relation), the first method is not effective. In order to use such knowledge, Glitter introduced several types of planning steps, and the programmer was sometimes required to assist in those steps. In terms of the original goal, Glitter has been fairly successful: In one example an unaided programmer was required to select 31 distinct transformation steps; with Glitter's assistance, all 31 were selected automatically, and the programmer was required to assist Glitter in only 6 of Glitter's 20 planning steps.

**Proust.** Proust is an experimental tutoring system designed to help novice programmers learn how to program (29). It has three main components: a collection of plans embodying alternative techniques for writing simple numeric programs, a surface analyzer that converts Pascal programs into the plan representation, and a technique for detecting small differences between two plans. Proust's strategy is to produce alternative (correct) programs from the specification and to find that which most closely matches the student's program. The differences between them are then assumed to be bugs in the student's program. As a simple example, consider the problem of computing the average value of the nonnegative elements of a list. The student program is:

```
Valid := 0;
Sum := 0;
for X in List do
  if X = 0
    then Valid := Valid + 1
    elseif X > 0
      then Sum := Sum + X;
Average := Sum/Valid.
```

Here, the student has forgotten to increment Valid in the **else** clause, a bug detected by comparing the program with a standard plan for using a variable as a counter.

**The  $\phi_0$  System.** The  $\phi_0$  ("phi-naught") system was actually an automatic programming system, but it had many of the characteristics which would be expected in a domain-specific knowledge-based programming assistant (4). The application domain involved numeric software used in the interpretation of data from measurements made in oil wells;  $\phi_0$  included a knowledge base of facts and relationships about the domain, a graphical interface through which the user could augment the knowledge base or specify programs using domain-specific concepts, an algebraic manipulator for solving systems of equations, and a translator for producing code from explicit equations. As a simple example, consider the problem of computing the relative volumes of the different minerals in a rock formation. A specification (in domain terms) for such a problem con-

sists of a list of the minerals and a list of the available measurements. Given domain knowledge about the way that the measurements depend on the minerals in the formation,  $\phi_0$  can express such a specification as a system of equations. For example, consider the problem of determining what fraction of a rock formation is solid and what fraction is fluid from a measurement of the speed of sound in the rock formation. Based on domain knowledge about the speed of sound in solid and fluid,  $\phi_0$  can express this specification as a system of equations:

$$\begin{aligned}\text{SpeedOfSound} &= 55 \cdot \text{VolumeOfSolid} \\ &\quad + 189 \cdot \text{VolumeOfFluid} \\ 1 &= \text{VolumeOfSolid} + \text{VolumeOfFluid}\end{aligned}$$

Then,  $\phi_0$  uses algebraic manipulation to solve the system:

$$\begin{aligned}\text{VolumeOfFluid} &= \frac{\text{SpeedOfSound} - 55}{134} \\ \text{VolumeOfSolid} &= 1 - \text{VolumeOfFluid}\end{aligned}$$

The resulting system is then translated into a sequence of assignment statements in the target language. Thus,  $\phi_0$  uses its knowledge of the domain and mathematics, as well as programming, to support programming for a special-purpose domain.

**Comparison.** All of the programming assistants described in this section are primarily concerned with programming in the small, but they have different intended users, support different tasks, and embody different types of knowledge. These differences are summarized in Table 1.

A variety of other experimental programming assistants have been developed. A representative sampling includes CHI (30), CIP (31), SETL (32), TAMPR (33), DRACO (34), and a data-structure-selection system (35).

### Future Possibilities

Predictions about future technological developments are always difficult to make, but it seems likely that practical and useful knowledge-based programming assistants will begin to appear within the next few years. Initially, the systems will probably support only isolated programming tasks, primarily

for programming in the small, and the systems' knowledge will be limited to general programming knowledge and not domain knowledge. The next round of systems will be more integrated, providing support for several tasks within a uniform framework. At the same time systems that embody limited forms of domain knowledge will begin to appear. Fully integrated, domain-specific knowledge-based programming assistants will begin to appear by the early 1990s, but they will be developed using specialized techniques. It is not likely that general techniques for developing integrated knowledge-based programming assistants will be well-understood until the mid-1990s. Somewhat paradoxically, it seems likely that sophisticated domain-specific automatic programming systems will become practical in limited situations before knowledge-based programming assistants, primarily because an automatic programming system for a restricted situation requires less knowledge than a programming assistant that must interact with the user in a more detailed and flexible way. In either situation, the support that seems to be furthest in the future involves the recording and retrieval of information about the decision history of a piece of software. In the meantime, however, it seems clear that knowledge-based programming assistants will, in some form, begin to provide substantial support for human programmers in the relatively near future. Perhaps the least predictable aspect of future knowledge-based programming assistants is the effect that they will have on the software engineering process itself—the only certainty is that the effect will be profound.

### Additional Sources of Information

For general discussions of these issues, see Refs. 36–39. For an extensive survey of program transformation systems, see Ref. 10. For collections of articles about general programming environments, see Refs. 40 and 41.

### BIBLIOGRAPHY

1. T. Teitelbaum and T. Reps, "The Cornell Program Synthesizer: A syntax-directed programming environment," *CACM* 24(9), 563–573 (September 1981), reprinted in Ref. 40.
2. W. Teitelman, Automated programmer: The programmer's assistant, in *Fall Joint Computer Conference*, Anaheim, CA, pp. 917–921, December 1972, reprinted in Ref. 40.
3. W. Teitelman, A Display-Oriented Programmer's Assistant, in D. Barstow, H. Shrobe, and E. Sandewall (eds.), *Interactive Programming Environments*, McGraw-Hill, New York, pp. 240–287, 1984.
4. D. Barstow, R. Duffey, S. Smoliar, and S. Vestal, An Automatic Programming System To Support an Experimental Science, *Sixth International Conference on Software Engineering*, Tokyo, Japan, September 1982, pp. 360–366.
5. W. Swartout, GIST English Generator, *Second National Conference on Artificial Intelligence*, Pittsburgh, PA, August 1982, pp. 404–409.
6. D. Cohen, A Forward Inference Engine to Aid in Understanding Specifications, *Fourth National Conference on Artificial Intelligence (AAAI)*, Austin, TX, August 1984, pp. 56–60.
7. D. Cohen, Symbolic Evaluation of the Gist Specification Language, *Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, August 1983, pp. 17–20.
8. C. Rich and H. Shrobe, "Initial report on a LISP programmer's apprentice," *IEEE Trans. Softw. Eng.* 4(6), 456–467 (November 1978), reprinted in Ref. 40.

**Table 1. Users, Tasks, and Types of Knowledge Utilized by Five Programming Assistants Described**

System	PA	DRA	Glitter	Proust	$\phi_0$
Intended user					
Expert programmer	x	x	x		
Novice programmer				x	
Domain expert					x
Programming task					
Specification					x
Implementation	x	x	x	x	x
Optimization		x	x		
Knowledge					
Programming	x	x	x	x	
Target language	x			x	x
Mathematics					x
Domain					x



9. S. Katz and R. Zimmerman, An Advisory System for Developing Data Representations, *Seventh International Conference on Artificial Intelligence*, Vancouver, B.C., August 1981, pp. 1030–1036.
10. H. Partsch and R. Steinbruggen, "Program transformation systems," *Comput. Surv.* **15**(3), 199–236 (September 1983).
11. Z. Manna and R. Waldinger, "A deductive approach to program synthesis," *ACM Trans. Program. Lang. Sys.* **2**(1), 90–121 (January 1980).
12. D. Smith, "Top-down synthesis of divide-and-conquer algorithms," *Artif. Intell. J.* **27**(1), 43–96 (September 1985).
13. D. Barstow, "An experiment in knowledge-based automatic programming," *Artif. Intell. J.* **12**(2), 73–119 (August 1979).
14. D. Barstow, *Knowledge-Based Program Construction*, Elsevier-North Holland, New York, 1979.
15. S. Gerhart, D. Musser, D. Thompson, D. Baker, R. Bates, R. Erickson, R. London, D. Taylor, and D. Wile, "An overview of AFFIRM: A specification and verification system," *Inform. Proc.* **80**, 343–347 (1980).
16. E. Kant, *Efficiency in Program Synthesis*, UMI Research, Ann Arbor, MI, 1981.
17. R. Paige, Supercompilers, in P. Pepper (ed.), *Program Transformation and Programming Environments*, Springer-Verlag, New York, pp. 331–340, 1984.
18. A. Borgida, S. Greenspan, and J. Mylopoulos. Knowledge representation as the basis for requirements specifications. *IEEE Computer* **18**(4), 82–91 (April 1985).
19. R. Mittermeir, Semantic Nets for Modeling the Requirements of Evolvable Systems, in J. Hagwood (ed.), *Evolutionary Information Systems*, North-Holland, New York, 1982.
20. D. Wile, "Program developments: Formal explanations of implementations," *CACM* **26**(11), 902–911 (November 1983).
21. E. Kant and D. Barstow, "The refinement paradigm: The interaction of coding and efficiency knowledge in program synthesis," *IEEE Trans. Sftwr. Eng.* **7**(5), 458–471 (September 1981), reprinted in Ref. 40.
22. R. Balzer, "Transformational programming: An example," *IEEE Trans. Sftwr. Eng.* **7**(1), 3–14 (January 1981).
23. T. Cheatham, J. Townley, and G. Holloway, A system for Program Refinement, *Fourth International Conference on Software Engineering*, Munich, FRG, September 1979, pp. 53–62, reprinted in Ref. 40.
24. B. Kedzierski, Knowledge-Based Communication and Management Support in a System Development Environment, Technical Report KES.U.83.3, Kestrel Institute, Palo Alto, CA, August 1983.
25. D. Barstow, "A perspective on automatic programming," *AI Mag.* **5**(1), 5–27 (Spring 1984).
26. C. Rich, Inspection Methods in Programming, Technical Report MIT/AI/TR-604, MIT, Cambridge, MA, August 1981.
27. R. Waters, "The Programmer's Apprentice: A session with KBEmacs," *IEEE Trans. Sftwr. Eng.* **11**(11), 1296–1320 (November 1985).
28. S. Fickas, "Automating the transformational development of software," *IEEE Trans. Sftwr. Eng.* **11**(11), 1268–1277 (November 1985).
29. W. Johnson and E. Soloway. Proust: knowledge-based program understanding. *IEEE Trans. Sftwr. Eng.* **11**(3), 267–275 (March 1985).
30. C. Green, J. Philips, J. Westfold, T. Pressburger, B. Kedzierski, S. Angerbrannndt, B. Mont-Reynaud, and S. Tappel, Research on Knowledge-Based Programming and Algorithm Design, Technical Report KES.U.81.2, Kestrel Institute, Palo Alto, CA, 1982.
31. F. Bauer and H. Wossner (eds.), *Algorithmic Language and Program Development*, Springer-Verlag, New York, 1982.
32. R. Dewar, A Grand, S.-C. Liu, and J. Schwartz, "Programming by refinement as exemplified by the SETL representation subsystem," *ACM Trans. Program. Lang. Sys.* **1**(1), 27–49 (July 1979).
33. J. Boyle, Software Adaptability and Program Transformation, in H. Freeman and P. Lewis (eds.), *Software Engineering*, Academic Press, New York, pp. 75–93, 1980.
34. J. Neighbors, Software Construction Using Components, Technical Report 160, University of California, Irvine, CA, 1980.
35. J. Low, Automatic data structure selection: An example and overview, *CACM* **21**(5), 376–385 (May 1978).
36. T. Winograd, Breaking the Complexity Barrier (Again), *Proceedings of the ACM SIGPLAN/SIGIR Interface Meeting on Programming Languages-Information Retrieval*, Gaithersburg, MD, November 1973, pp. 13–30, reprinted in Ref. 40.
37. R. Balzer, T. Cheatham, and C. Green, "Software technology in the 1990's: Using a new Paradigm," *IEEE Comput.* **16**(11), 39–45 (November 1983).
38. D. Barstow and H. Shrobe, From Interactive to Intelligent Programming Environments, in D. Barstow, H. Shrobe, and E. Sandewall (eds.), *Interactive Programming Environments*, McGraw-Hill, New York, pp. 558–570, 1984.
39. D. Barstow, The Role of Artificial Intelligence in Programming Environments, *Workshop on Software Engineering Environments for Programming-in-the-Large*, Harwichport, MA, June 1985.
40. H. Hünke (ed.), *Software Engineering Environments*. Elsevier-North Holland, New York, 1981.
41. D. Barstow, H. Shrobe, and E. Sandewall (eds.), *Interactive Programming Environments*, McGraw-Hill, New York, 1984.

D. BARSTOW  
Schlumberger-Doll Research

## PROGRAMMING ENVIRONMENTS

The software technology of AI is examined, beginning with a discussion of the nature of AI programming and the ways in which it differs from conventional software engineering. The characteristic programming tools that have evolved in response to these different needs are then reviewed, followed by a more detailed examination of their use in the INTERLISP programming environment. Finally, the advantages of AI programming technology for exploratory development in other (non-AI) domains are considered, ending with some speculation as to their possible future commercial evolution.

The software technologies used to develop AI programs—the programming languages, techniques, and environments that support them—are all very distinctive. Like all tools in all crafts, their characteristic form is the product of a long evolution in response to the changing functional demands of their task, in this case the development of computer programs that exhibit "intelligence." Thus, although one can argue their merits in the abstract, the entry begins by looking, from a software engineering perspective, at the programming problem with which these tools were designed to cope. Looked at from this perspective, it turns out that there are some characteristics of the AI programming task that set it apart from conventional programming in rather dramatic ways that have decisively shaped the tools used to do it.

**AI Programming.** The essential difficulty of the AI programming task lies implicit in the term "artificial intelligence."

Although a philosopher could make much of the word "artificial" (e.g., as Simon did in Ref. 1), it is *operationally* relatively benign, implying little more than the use of digital computers. However, one does not have to be a philosopher to realize the problems inherent in "intelligence." Minsky once characterized "intelligence" as an attribute used principally to describe intellectual behavior that people admire but do not understand. The iconoclasm of this remark should not be allowed to obscure its insight. To a large extent, the label "intelligence" says a great deal more about the state of ignorance of the labeler than it does about the phenomena being described. For example, at a certain age children will claim that long division requires "intelligence" because the "smart kids" can do it and the others cannot. Adults know better, they know what the trick is. Adults reserve "intelligence" to describe behavior that *they* do not understand, like medical diagnosis (unless one is doctor), or a three-year-old's grasp of language (once one understands that one's naive theories of language are completely ineffectual).

The implication for AI programming of this unsettling observation is that, if you are building a program which you are in any way tempted to describe as "intelligent," you are building a program that you do not completely understand. Your own (weak) description tells you that you do not entirely know what you are doing. And if by any chance you do know what you are doing (or if you find out along the way), it will not be AI any more because by that stage you will have a far more precise description of what you are working on than "intelligence." Some topics have actually reclassified themselves out of AI for just this reason. For example, symbolic mathematics was once one of the classic areas of AI. Now, however, enough is known about this problem that it has become a subfield of algorithm design and analysis. Symbolic mathematics is no longer AI because it is understood too well.

Perhaps coincidentally, AI programming is usually done by individuals or very small groups (often of captive graduate students). It is easy to see why this might be adaptive. If you do not understand exactly what you are doing, you will probably start out in the wrong direction. It will therefore be important to be able to change direction quickly. That mitigates against large programming teams, because, like all large groups, they just take too long to change direction. In any event, the large team would probably be unmanageable given the lack of a detailed specification since there would be no effective way to subdivide the problem so they could all work effectively.

None of this would be very important were it not that AI programs tend to be big. Whether the complexity is in the data, as in knowledge-base systems, or in the program itself, as in inference-based systems (see Inference), the net effect is the same. A small implementation team attacking a large programming problem with considerable uncertainty about the final design will be out of control. Being out of control is standard operating procedure in AI programming. Not because the programmers are bad (in fact, they are usually very good) but because the size and difficulty of the problems that they are dealing with always threaten to overwhelm them. Indeed, in many ways (perhaps disappointingly), the research boundary of AI tends to be a direct reflection of how complicated a program can be sustained by a small group at any given point in time.

**Programming under Uncertainty.** The effect of this "complexity standoff" is that a tremendous amount of effort has been

invested within AI to build programming technologies to help people cope with the problem of programming under uncertainty—programming when you do not know what you are doing. Central to all of these approaches is the observation that if you do not know what you are doing, an excellent way to find out is to try some experiments, to explore the problem area by implementing some tentative designs and seeing what works. But any such exploratory orientation places an enormous premium on making it easy to change your mind. Unless you are lucky, you usually do not get the right answer the first time you try something new. So you must expect to be in a continual state of tearing your code up in response to the latest failure and rebuilding it until you converge on a solution.

Note that, at this level, the argument is completely independent of one's choice of programming formalism (although it is argued below that it constrains that formalism in certain interesting ways). Whether one is using a programming language such as LISP, an expert-systems shell, or a domain-specific language, such as a natural-language grammar notation, an essential aspect of most (if not all) AI problems is that one does not know how to solve the problem when one starts and that one learns by studying the (faulty) behavior of one's partial solutions. This is quite different from a simple willingness to admit the possibility of mistakes. It is instead the acceptance of a very different process of programming in which making "mistakes" and learning from them plays a key role. Whether it be LISP or PROLOG, expert-systems shell, or natural-language grammar, the environments in which AI programming is done must at least accept, if not actively support, the fundamentally exploratory nature of this process to have any true match to the needs of their users.

**Conventional Programming Technology.** Notice how completely orthogonal this is to conventional programming methodology. There, one begins by writing down the design, checking it until one is sure it is right, and then constraining the coding process to follow that design so that one implements exactly what the design specifies and does not wander off in some unpredictable direction. That is absolutely the correct thing to do when one has confidence in one's design. But that is absolutely not what is wanted here. One wants to decide what is wanted at the *end* of the implementation process, not at the beginning when one knows almost nothing about the problem. The classical approach only works when you have perfect foresight, which one most certainly does not have in AI. As a result, most conventional programming methodologies can be (and are, in AI practice) discarded immediately. They do not work. They do not work because their objectives are completely different from the objectives of AI programming. Conventional programming methodologies try to limit exploration and enforce adherence to a known valid design. AI programming accepts the need to explore a problem in order to find out what works.

Not only are conventional programming techniques ineffective for AI development but they turn out to be absolutely counterproductive. A large programming project with uncertain or changing specifications is a particularly deadly combination for conventional programming methodology since this methodology will inevitably lead the project into a situation with which it cannot cope. The evolution of the debacle is instructive. The first step is the development of a detailed specification. The difficulty of doing this for AI problems is discussed above, so this process will not be easy, but eventually something has to be settled on so implementation can

begin. A conventional implementation team would then take this specification, partition it, define a module structure that reflects this partitioning, freeze the interfaces between them, and repeat this process until the problem has been divided into a large number of small, easily understood and easily implementable pieces. Control over the implementation process is achieved by the imposition of structure, which is then enforced by a variety of management practices and software tools.

Since the specification, and thus the module structuring, is considered fixed, one of the most effective classical methods for enforcing it is the use of redundant descriptions and consistency checking—thus the importance of techniques such as interface descriptions and static type checking, which require multiple statements of various aspects of the design in the program text in order to allow mechanical consistency checks to ensure that each piece of the system remains consistent with the rest. In a well-executed conventional implementation project, a great deal of internal rigidity is built into the system in this way in the interests of ensuring its orderly development.

The problems emerge very late in the implementation when it is realized that, not just superficial, but radical changes are required in the design. From the point of view of conventional programming practice, this indicates a failure at specification time. The implementors should have been more persistent in working out the implications of the design before starting to code. And in a situation where conventional programming is appropriate, this is probably the correct diagnosis. Many ordinary implementation exercises are brought to ruin by insufficient attention having been paid to getting the consequences of the design fully understood. But that is not the case here. If the design had been completely understood, it would in all probability never have been implemented (it would not be “AI”). No, the design was probably implemented precisely *because* not all of its consequences were understood. It should come as no surprise that one (typically) does not like all of them.

Now, however, one must rework the system to avoid the behavior that one does not like. But that presents a situation that conventional programming methodology simply refuses to acknowledge (except as something to avoid). As a result, conventional programming tools and methods are suddenly of limited effectiveness, if not actually counterproductive. The redundant descriptions and imposed structure that were so effective in constraining the program to follow the old specification have lost none of their efficacy—they still constrain the program to follow the old specification. And they are difficult to change. The whole point of redundancy is to protect the design from a single (unintentional) change. But it is equally well protected against a single intentional change. Thus, all the changes have to be made everywhere. (And, since this should never happen, there is no methodology to guide our programming tools to assist this process.) Of course, if the change is small (as it “should” be), there is no particular problem. But if it is large, so that it cuts across the module structure, the implementor(s) find that they literally have to fight their way out of their previous design.

Still, no major problem exists if that is the end of the matter. But the behavior of the new system is just as likely to be not right, which will suggest yet another change. And so on. After a few iterations of this, not only are the implementors exhausted but also the repeated assaults on the module structure have likely left it looking like spaghetti. It still gets in the way (firewalls are just as impenetrable if laid out at random as

they are when laid out straight) but has long ceased to be of any use to anyone except to remind them of the sorry history. Increasingly, it is actively subverted by programmers whose patience is running thin. Even were the design suddenly to stabilize, all the seeds have now been sown for an implementation disaster as well.

The alternative to this kind of predictable disaster is not to abandon structured design for programming projects that are, or can be made, well-defined. Instead, one should recognize that AI applications are best thought of as design problems rather than as implementation projects. These problems require programming systems that allow the design to emerge from experimentation with the program so that design and program develop together. It is this imperative, more than anything else, that has shaped the various AI programming systems.

**AI Programming Systems.** In contrast to conventional programming technologies, which restrain the programmer in the interests of orderly development, AI programming systems try to amplify the programmer in the interest of maximizing his range. The exploration that is required to develop an AI program can require small numbers of programmers to make massive transformations repeatedly to very large amounts of code. Such programmers need programming power tools of considerable capacity or they will simply be buried in detail. So, like an amplifier, their programming tools must magnify their necessarily limited energy and minimize extraneous activities that would otherwise compete for their attention.

The exploratory AI programming technology that results has three distinct components—languages, techniques, and environments—each providing the programmer with a distinct source of leverage.

### Languages

The programming languages of AI are covered elsewhere in this encyclopedia, individually and in detail. However, despite this, and despite the fact that they differ considerably, they have some elements in common which are key to understanding the distinctive needs of AI programmers.

One key principle of AI programming languages is to allow the programmer to minimize and defer commitment as long as possible. A decision that has not been cast into code does not have to be recast when it is changed. Thus, the longer one can carry such a decision implicitly, the better. In some cases a decision can be deferred until later during program development; in others it can wait until program compilation (so no code needs to be changed when the decision is changed); and in some cases the decision can be deferred all the way until run time (so it can be changed dynamically, while the program is running). For one example, it is rarely advantageous for the programmer to predefine the exact usage of storage and AI languages are as one in supporting automatic storage allocation and garbage collection. For another, unlike most conventional languages, AI languages are typically untyped (any variable can hold any object) to allow easy experimentation with the application's type structure. For yet another, these languages often allow one to defer commitment about what any given operation means by passing the procedure that does it as an argument, constructing a new one and returning it as a value, or even redefining a procedure on the fly, while the program that uses it is running. In the limit, using the technique of object-oriented programming (see Languages, object-

oriented), one can actually bury the procedures associated with a piece of data inside its value, so that each object carries with it its own ideas of how to respond to certain operations. In summary, these languages provide a wide variety of techniques for deferring commitment, for making experimentation easier by minimizing how much you have to do when you back out and change your mind.

Any (even passing) discussion of AI programming languages would be seriously incomplete without mentioning the role of LISP. The majestic domination of AI programming by LISP and LISP dialects over the past 20 years is quite remarkable. Unlike the economic motivation (large amounts of existing code) that has kept languages like FORTRAN alive across a similar sweep of computing history, LISP has survived and prospered in the AI community because it has embedded within it some design choices that are a singularly good match to AI.

Arguably the most important of these is the ability to support embedded languages. In the case of LISP, this is enabled by LISP's simple (easily described within LISP) representation for LISP programs. This makes it straightforward to write LISP programs that can manipulate other LISP programs as data—reading them to determine what they do, making changes to them, creating new code from them. This apparently arcane property has enabled three fundamental innovations. First, it enabled experimentation on and extension of the LISP language to take place within LISP itself. Although one tends to talk about “LISP” as if it were a unitary language, it is actually a rapidly evolving collection of dialects that have absorbed (and continue to absorb) many ideas from other languages because of this powerful bootstrapping ability. Second, it enabled the programming technique of embedded languages, discussed below. Third, the ability of LISP programs to examine and modify other LISP programs facilitated the development of many of the programming environment tools that are discussed below.

### Programming Techniques

The second component of AI software technology is a wide range of programming techniques that have been developed to facilitate AI programming. Some of these, e.g., automatic backtracking (qv), have been so significant that they have been incorporated into specialized languages and environments. Others of note include data-directed programming, rule-based systems (qv), constraint propagation (see Constraint satisfaction), truth maintenance, and various systems for knowledge representation (qv) (for a more complete treatment see also Ref. 2).

A crucial enabling technology for many of these techniques has been to use the linguistic flexibility of LISP to take an application-specific concept and embed it in LISP, producing a new language that is crafted directly for the application as it is currently understood. For example, one can take the problem-solving strategy of automatic backtracking and embed that in LISP, with its own operations, syntax, and semantics, to form a new language (in this case, Planner) which is all of LISP, plus the fundamental primitives of (and the environmental support for) that particular problem-solving strategy. This is important because it allows you to start thinking in those terms. You have an idea about what is important, you build your idea into an extended language, and now you program in terms of your idea. Language embedding within LISP has

been used to develop a whole series of specialized sublanguages (e.g., Planner, CONNIVER (qv), QA4, etc.) each of which has been used to explore the AI problem-solving paradigms that it encapsulated.

It should come as no surprise that despite the fact that they were developed in the specialized context of AI, many AI programming techniques turn out to have considerable general programming applicability. Although the problem to which they are directed may have arisen first, or more acutely, in an AI context, there is often little about them that is particularly idiosyncratic to AI applications. For example, rule-based systems, although central to the “expert-system” (qv) style of AI application, are a quite general programming technique for structuring large systems so that small pieces of code can be incrementally added without having to understand the entire control structure. This is an enormously powerful programming idea that says little if anything about the nature of intelligence, per se, but has a lot to do with the task of managing large programs that need frequent, local changes.

### Programming Tools

Impressive though the AI programming languages and techniques are, they only speak to half the problem. Their key feature is that they encourage the programmer to defer commitments as long as possible in the interests of preserving flexibility. In some cases, such as automatic storage management, the programming system has to absorb a significant burden from the programmer in order to allow this freedom from commitment. However, the programmer still has the significant problem of keeping track of a large, rapidly evolving system. Further, in the interests of constraint deferral, many of the language features, such as static checking of redundant specifications, that might have supported conventional programming methods have been removed. How is the programmer to regain control?

The essential element is an integrated programming environment that provides help where it can and automates the control and bookkeeping functions that are the greatest drain on a programmer during rapid system development. An environment like this can actively assist the programmer by

- deriving and maintaining information about the user's program as they change;
- providing ways to view this information and use it to make systematic transformations to the evolving system;
- providing active “agents” that notice events that the programmer should pay attention to, take action on them, propose changes, and automatically take care of routine details (like cleaning up after a session); and
- supporting ad hoc performance engineering of those parts of the system that have achieved design stability so that one does not pay a performance penalty for one's initial uncertainty about the design.

Each of these is discussed in the abstract as well as their role in one environment.

**Interactive Interface.** Programming tools achieve their effectiveness in two quite different ways. Some are simply effective viewers into the user's program and its state. Such tools permit one to find information quickly, display it effectively, and modify it easily. Tools of this form include data-value

inspectors (which allow a user to look at and modify the internal structure of an object), specialized editors for code and data objects, and a variety of instrumentation, break, and tracing packages.

Especially when coupled with a large-format display and a pointing device like a "mouse," which allows very-high-bandwidth communication with the user, such viewers are very effective programming tools and enable a virtually endless variety of ways to display the state of the user's environment. For example, the system might present storage utilization in a continuously adjusting bar graph or display a complicated data structure as a network of nodes and arcs and allow the user to edit this representation directly. Complicated commands for specifying objects can often be obviated by the ability to display the data structure and have the user point at the appropriate place. Similarly, the choice between concise (easily typed) but esoteric command names as opposed to longer (to type), more self-explanatory ones becomes academic when commands can be invoked via menus.

In addition to programming tools, personal machine-programming environments also allow the standard features of a professional work station, such as document preparation, file management, and electronic mail, to be provided within the programming environment itself. Not only are these facilities just as effective in enhancing the productivity of programmers as they are for other professionals but also their integration into the programming environment allows them to be used at any time during programming. Thus, a programmer who has encountered a bug can send a message reporting it while remaining within the debugger, perhaps including in the message some information, like a backtrace, obtained from the dynamic context.

**Knowledge-Base Tools.** The other type of programming tool is knowledge-based. Viewer-based tools, such as program text editor, can operate effectively with a very limited "understanding" of the material with which they deal. By contrast, knowledge-base tools must know a significant amount about the content of a user's program and the context in which it operates. Even a very shallow analysis of a set of programs (e.g., which programs call which other ones) can support a variety of effective programming tools. A program browser allows a programmer to track dependencies between the different parts of a program by presenting easy-to-read summaries that can be further expanded interactively.

Other pieces of knowledge can be used to provide a variety of automatic programming support services. One key piece of knowledge is where in the file system to find each programs' source code. This knowledge enables the user, whenever he or she decides to edit a particular fragment of code, to ask for it by name (or some other meaningful description, like how it is used) and have the system find it, bring it to the user (in some appropriate editor), and then put it away when the edit is done. During a sequence of edits the system can keep track of what has changed so it can automatically update and recompile the corresponding files afterward.

Deeper analysis allows more sophisticated facilities. For example, a static-program analyzer that can read and analyze user programs can be used to build a database about them that can be used by a wide variety of tools to answer complex queries (which may require significant reasoning, such as computing the transitive closure of some property), make systematic changes under program control (e.g., "Find me all the places

where I've done such-and-such and change them to do so-and-so"), or check for a variety of inconsistent usage errors. Note that this type of usage is critically dependent on the ability of the programming support tools to look at the user's code as it is written or changed and derive the enabling information. If the user has to supply it by hand, not only does this add to the total burden on the programmer but also the information is highly unlikely to be available (or reliably up to date) when it is needed.

**Tool Integration.** The integration of the programming support tools provide yet another level of power. For example, if the system "notifies" whenever a program fragment is changed (by the editor or by redefinition), the program analyzer can then be informed that any existing analysis is invalid, so that incorrect answers are not given on the basis of old information. The same mechanism can be used to notify the program-management subsystem (and eventually the programmer, at session end) that the corresponding file needs to be updated. In addition, if the system remembers the previous state of the program, at any subsequent time the programmer can undo the change and retreat (in which case, of course, all the dependent changes and notifications must also be undone). This level of cooperation between tools not only provides immense power to the programmer but it also removes a level of tasks that would otherwise have to be explicitly managed. As a result, the programmer has more attention to pay to exploring the design and managing the program complexity that is inevitable during fast track development.

Of course, in a personal-computing environment the knowledge-based facilities will all have elaborate interactive graphic interfaces that provide specialized viewers and information-management systems to support them. This will greatly enhance their usefulness. Also, in a personal-computing environment the processing for many of these tools, e.g., the process of maintaining the database about the programs under development, can be done continually in a background mode while the user is thinking (other facilities, such as an incremental garbage collector, can also operate in this fashion). This can be a qualitative change in the effectiveness of many programming tools.

**Contraction.** A key, but often neglected, issue in AI programming is program contraction. The development of an AI program is "design-limited," so that is where the initial effort has to go. Consequently, the program is often both inefficient and inelegant when it first seems to have achieved functional acceptability. If this initial exploration is adequate by itself, this is of limited concern. However, for many AI systems their application to real problems are very large computations. Thus, the ability to make these programs efficient is of central concern because without it they would never be run on any interesting problems. So, although AI environments are often used to write code very quickly without any concern for efficiency, they must also support turning out an efficient implementation of the final result.

In the case of AI commercial applications, the importance of being able to make this post hoc optimization cannot be over-emphasized. Without it, one's initial development will always be considered a "toy," and the pressure to abandon the exploratory environment and start implementing in a "real" one will be overwhelming. Once that move is made (and it is always made too soon), exploration comes to an end. The program-



ming system must meet two requirements if it is to support an acceptable level of program contraction.

First, the architecture has to permit efficient implementations. For example, the obligatory automatic storage manager must either be so efficient that it imposes negligible overhead or it must allow the user to circumvent it (e.g., to allocate storage statically) when and where the design has stabilized enough to make this optimization possible. Traditionally, this need was met mainly by circumvention. However, more recently, the specialized LISP machines (qv) have introduced a variety of techniques for high-performance implementation of the operations that are relatively slow in a traditional LISP implementation. One measure of their success is that they have been able to implement these systems entirely in LISP, including all of the performance-critical system code such as the operating system, display software, device handlers, etc.

Second, as the performance engineering of a large system is almost as difficult as its initial construction, the environment must provide performance engineering tools, just like it provides design tools. These include good instrumentation, a good optimizing compiler, program-manipulation tools (including, at the very least, full-functionality compiler macros), and the ability to add declarative information where necessary to guide the program transformation. Note that, usually, performance engineering takes place not as a single "post-functionality optimization phase" but as a continuous activity throughout the development as different parts of the system reach design stability and are observed to be performance-critical. This "progressive constraint" approach, the incremental addition of constraints as and when they are discovered and found important, is a key AI software methodology.

## INTERLISP

The principal components of an AI software environment are reviewed in the abstract above, and the form they take in the INTERLISP programming environment is now considered. INTERLISP is a programming environment based on a dialect of the LISP language, which has been used to develop and implement a wide variety of large application systems, primarily in the area of AI (3). It is a useful example, not only because it is one of the most widely used programming environments for AI but also because many of the tools just discussed were first implemented within INTERLISP [although, as Teitelman and Masinter (4) make clear, their true power did not become clear until very late in their development].

INTERLISP provides an extensive set of user facilities, including syntax extension, uniform error handling, automatic error correction, an integrated structure-based editor, a compiler and interpreter, a debugger, and a filing system. However, from its inception, the focus of INTERLISP has been to construct a programming "environment" rather than a programming language. An early paper on INTERLISP in Ref. 5 states:

*The programmer's environment influences, and to a large extent determines, what sort of problems he can (and will want to) tackle, how far he can go, and how fast. If the environment is cooperative and helpful (the anthropomorphism is deliberate), the programmer can be more ambitious and productive. If not, he will spend most of his time and energy fighting a system that at times seems bent on frustrating his best efforts.*

Environments that are "cooperative and helpful" imply a willingness to "let the machine do it." INTERLISP is based on a conscious decision to expend computer resources to enhance the performance of the programmers who use it. Occasionally, this has produced tools whose demands for computer resources were ahead of their time, but more often the capacity has become available for tools whose utility was clear.

Historically, INTERLISP has exhibited "a somewhat chaotic growth pattern and a style sometimes characterized as baroque" (4). The reason is that INTERLISP was not designed but evolved in response to the experience of its users. As Sandewall (6) points out:

*The task of designing interactive programming systems is hard because there is no way to avoid complexity in such systems. . . . The only applicable research method is to accumulate experience by implementing a system, synthesize the experience, think for a while, and start over.*

Much of the design difficulty comes from the inherent complexity of the interactions among the sophisticated tools, which makes it very difficult to design simple interfaces for them in advance. Usually, unification and simplification of designs of this complexity come only after considerable experience. Also, many of the low-level facilities of INTERLISP were only added after their necessity to support the higher-level tools became clear. For example, INTERLISP provides a uniform error handler and permits accessing the control stack at an unusually detailed level. Capabilities such as these have been added to INTERLISP precisely in order to support its sophisticated debugging and error facilities. They were not there to be used before these tools were designed.

In fact, the design of a complex, knowledge-based programming system like INTERLISP is itself an exploratory programming task of the first order. Thus, it should come as no surprise that INTERLISP is implemented in itself. That each version of INTERLISP has supported both the rapid prototyping and the eventual production implementation of its successor is a pleasing confirmation of the power and efficacy of the environment for fast-track development.

**Interactive Interface.** The INTERLISP user sees the programming environment through a collection of display windows, each of which corresponds to a different task or context. The user can manipulate the windows, or the contents of a particular window, by a combination of keyboard inputs and pointing operations. The technique of using different windows for different tasks makes it easy for the user to manage several simultaneous tasks and contexts, e.g., defining programs, testing programs, editing, asking the system for assistance, and sending and receiving messages. It also eliminates the need for any explicit context switch when switching between tasks or programming tools. Thus, having called the editor from inside the debugger, the user can examine the current runtime state or ask MASTERSCOPE a question without losing the context of the editing session.

**File Package.** Interactive-program development consists alternately of testing pieces of programs and editing them to correct errors discovered during the tests and/or to extend the program. Throughout this process the primary copy of the program (the copy that is changed during editing operations) re-



sides in the INTERLISP system as a data structure; editing is performed by modifying this data structure, and testing is performed by evaluating it using the interpreter. For this reason, INTERLISP is called a residential system (6).

In a residential system one must be able to take the data structures that represent procedures and save them on external files. The INTERLISP file package is a set of programs, conventions, and interfaces with other system packages that facilitate this by automating the bookkeeping necessary to maintain a large system consisting of many source files and their compiled counterparts. The file package removes from the user the burden of keeping track of where things are and what things have changed. On request, it automatically retrieves any required program object if it is not already in the user's working environment. The file package also keeps track of which objects have been in some way modified and need to be written out to files, which files have been updated but still need to be recompiled, etc.

The file package operates automatically and behind the scenes by means of "hooks" into many parts of the system. For example, since INTERLISP's editors record which objects the user has changed, the file package can provide a function (CLEANUP) that enumerates all the user's files and updates those containing objects that have changed. An automatic warning is issued if an object not associated with any file has been changed or newly defined and the user is invited to specify where it should go. The file package supports the abstraction that the user is manipulating his program directly and that the file is merely one particular external representation of a collection of program pieces. During the session the user manipulates the pieces with a variety of editors and tools, occasionally saving what has been done by calling CLEANUP. The process of maintaining source-code consistency is entirely automated.

**MASTERSCOPE.** As a program grows larger and larger, it becomes increasingly difficult to predict the effect of a proposed change. It also gets difficult to effect a pervasive change, e.g., to change the calling convention of a procedure and be sure that all of the places in other programs that use it have been found and modified. MASTERSCOPE is an interactive subsystem that addresses this problem by analyzing user programs to provide the user with information about which functions are called, how and where variables are bound, set, or referenced, which functions use particular record declarations, etc. Not only does MASTERSCOPE know about the built-in LISP functions but it also provides a "template" language to allow programmers to define the interpretation of new forms added to the language.

MASTERSCOPE maintains a database of the results of the analyses it performs. The user can either interrogate the database explicitly (e.g., WHO USES FOO FREELY) or automatically call the editor on all the places that satisfy certain conditions (e.g., EDIT WHERE ANY FUNCTION USES THE RECORD DICTENTRY). The utility of MASTERSCOPE depends on the fact that the burden of remembering what has changed, and therefore needs reanalysis, is not carried by the user but is carried out automatically behind the scenes.

**DWIM.** One of the things that programmers do is make mistakes. Often these mistakes are simple typos and semantic confusions that any programmer looking over another's should

der would pick up and correct with little or no context. Clearly, a "helpful" environment should also provide the programmer this level of assistance.

The most visible part of DWIM (7) is the spelling corrector, which is invoked from many places in the system, including the file package and the editor as well as the LISP interpreter. When an unrecognized item is encountered in any of these contexts, the spelling corrector is invoked to find the closest match within a list of relevant items. If an edit command is misspelled, e.g., the list of valid edit commands is searched; if the name of a function is misspelled, the corrector scans a list of the functions the user has recently been working with. If the spelling correction is successful, the cause of the error is also repaired, so subsequent corrections will not be necessary. For example, when DWIM corrects a user's misspelled function name in one of his programs, it actually modifies the user's program to contain the correct spelling (and notifies the file package of the change).

Although most users think of DWIM as a single identifiable package, it embodies a pervasive philosophy of user-interface design: At the user-interface level system facilities should make reasonable interpretations when given unrecognized input. Spelling correction is only one example of such an interpretation. DWIM is an embodiment of the idea that the user is interacting with an agent who attempts to interpret the user's request "appropriately." That does not include being stopped and forced to correct oneself or give additional information when the correction or information is obvious.

**Programmer's Assistant.** Another corollary of the same idea is that the user should not find the system to be a passive partner that merely responds to each input and waits for the next but should be interacting with an active intermediary that maintains its side of the "conversation." The programmer's assistant (8) records each user input, along with a description of its side effects and results. The user can then give commands that refer to previous ones. For example, the REDO command allows the user to repeat a particular operation or set of operations; the FIX command allows the user to invoke the editor on a specified operation and then to reexecute the "fixed" version; the UNDO command cancels the effect of a specified operation. In addition to the obvious use of recovering information lost through erroneous operations, UNDO is often used to flip back and forth between two states. For example, one might make some changes to a program and/or data structures, run an experiment, undo the changes, rerun the experiment, undo the UNDO, etc. (see Programming assistants).

**Extensibility.** Virtually all of the facilities described can be redefined or extended to support new applications. Some of these extensions have been provided by simple macros or open-ended command interpreters. However, many extensions are not expressible in these terms because they are triggered, not by the appearance of a particular token, but by the existence of a more general condition. INTERLISP provides a variety of interfaces that allow the programmer to specify the response to various conditions, such as dealing with some object/expression/command that a particular facility does not recognize.

For example, the DWIM facility, which corrects spelling errors encountered in running programs and expressions, is implemented via an extension to the LISP interpreter. When-

ever the INTERLISP interpreter encounters a situation in which it is going to generate an error (e.g., an undefined function or variable), the interpreter instead calls a known function that tries to correct the spelling of the undefined function or variable, according to the context in which the error occurred. This public entry provides a capability that is also available for other uses. For example, one application redefined that function to send error messages to the implementor, not the user, of the application, via computer mail.

### AI Tools and Non-AI Problems

The software tools of AI have evolved to handle programming situations where there is significant design instability and limited ability to predict how effective any given solution will be. They have shown themselves to be remarkably effective in helping programmers to explore these complex design spaces rapidly and converge on an appropriate design.

The relevance of this technology outside of AI should be clear—coping with uncertainty is an increasingly common programming problem (9). Many ordinary computing applications, although not in any sense AI, turn out on inspection to have all the symptoms of programming under uncertainty, whether it be for reasons of design complexity, idiosyncratic human interface issues, or completely exogenous factors that inhibit design stability. The easy applications, the places where one can find a well-defined procedure that can routinely be automated, are getting hard to find. Increasingly, one is confronted with situations where there is not a well-defined procedure, where no one knows what is really needed. These are ideal matches to the strengths of AI software technology. That technology will not make the exogenous factors that contribute to the uncertainty disappear, but it at least holds out the promise of being able to adapt to them fast enough so that the uncertainty can be mapped and the range of design possibilities understood.

In retrospect, it is becoming clear that applications lie along a spectrum of uncertainty, between the very well understood ones (which are mainly done now) and the extremely ill-understood ones (which are mainly not, because no one knows how to start). AI occupies the latter extreme of that spectrum. AI is programming under terminal uncertainty, programming under uncertainty by self-definition. Conventional data processing occupies the other end of that spectrum. Until recently there has only been one cost-effective application development technology, one that assumes that everything is fixed and certain. As a result, the application spectrum has been dichotomized, and everything that was considered doable (which did not include AI) has been deemed to be completely understood because the application methodology demanded it. The availability of an effective technology for dealing with design uncertainty removes the need for such an artificial barrier and suggests that we revisit many applications that have resisted conventional development methods. For example, an enormous variety of specialized, professional information-support systems now seem to be realizable given AI's software tools to develop them.

The first demonstrations of this are now taking place in the construction of the support environments being designed for the first commercial AI applications. An AI application (e.g., an expert advisor on subject x) is of little practical use in isolation. The AI functionality must be embedded within a system that provides or interfaces to a significant information-management function (e.g., a database, analysis, or filing sys-

tem) for the application area. Quite often, however, effective interactive information utilities have never been built for the intended users of these AI systems. Equally frequently, they have been built but turn out to be ponderous compromises limited by the inability of a team using conventional technology to track the needs of their users. As a result, the supporting information environments for many of the first AI applications, which will take advantage of the prototyping power of the AI software-development technology, are likely to have a revolutionary impact on their users. And since the ability to build highly customized information-management systems is itself an application opportunity of enormous leverage, it should not take long for the power of this software technology to be recognized and put into much wider use. Whether this results in the incorporation of many of the ideas of the AI environments into more conventional software practice remains to be seen.

To those accustomed to the precise, structured methods of conventional system development, AI software development techniques will seem messy, inelegant, and unsatisfying. But it is a question of congruence: Precision and inflexibility are just as disfunctional in novel, uncertain situations as procrastination and vacillation are in familiar, well-defined ones. For AI or any problem that shares it levels of uncertainty, the efficacy of the AI software-development techniques has been clearly demonstrated.

### BIBLIOGRAPHY

1. H. Simon, *Sciences of the Artificial*, Addison-Wesley, Reading, MA, 1972.
2. E. Charniak et al., *Artificial Intelligence Programming*, Lawrence Erlbaum, Hillsdale NJ, 1980.
3. *The Interlisp Reference Manual*, 3 vols. Xerox Artificial Intelligence Systems, Pasadena, CA, 1985.
4. W. Teitelman and L. Masinter, "The Interlisp programming environment," *Computer* 14(4), 25–34 (April 1981).
5. W. Teitelman, Toward a Programming Laboratory, *First International Joint Conference on Artificial Intelligence*, Washington, DC, May 1969, pp. 1–8.
6. E. Sandewall, "Programming in the interactive environment: the Lisp experience," *CACM* 10(1), 35–71 (March 1978).
7. W. Teitelman, "Do what I mean," *Comput. Automat.*, (April 1972).
8. W. Teitelman, Automated Programming: The Programmer's Assistant, *Proceedings of the Fall Joint Computer Conference*, AFIPS, Reston, VA, 917–922, 1972.
9. B. Sheil, "Power tools for programmers," *Datamation* 29(4), 131–144 (February 1983).

B. SHEIL

Xerox Artificial Intelligence Systems

**PROGRAMMING LANGUAGES.** See EMYCIN; Logic programming; MICROPLANNER; Languages, object-oriented; LISP; Loops; OPS-5; Planner; POP-2; PROLOG; Smalltalk; Snobol-4.

### PROLOG

A computer-programming language based on the notion of logic programming (qv), PROLOG was written around 1970 by A. Colmerauer at the University of Marseille; it is the lan-

guage used by the Japanese fifth-generation computer project (see Computer systems) (see W. F. Clocksin and C. S. Mellish, *Programming in Prolog*, Springer-Verlag, Berlin, Heidelberg, 1981).

M. TAIE  
SUNY at Buffalo

**PROPELLING MECHANISMS.** See Robotics.

**PROPOSITIONAL CALCULUS.** See Logic, propositional.

## PROSPECTOR

An expert system (qv) that aids geologists to interpret mineral data and to find ore deposits from geological data, PROSPECTOR was written in 1979 by Duda, Hart, and others at SRI International in California (see Rule-based systems) (see R. O. Duda, J. G. Gaschnig, and P. E. Hart, Model Design in the PROSPECTOR Consultant System for Mineral Exploration, in D. Michie, (ed.), *Expert Systems in the Micro-Electronic Age*, Edinburgh University Press, Edinburgh, U.K., pp. 153-167, 1979).

M. TAIE  
SUNY at Buffalo

## PROSTHESES

There are two truths in AI and robotics (qv) today: the first is that most people in research and development would like their creations to match or exceed human cognitive, sensory, mobility, and manipulative capabilities; the second truth is that none of these machine functions is yet comparable to its human counterpart. The validity of this assertion is evident when there is an attempt to have a robot perform the routine daily living task of cooking a meal. However, there is real promise in the utility of having machines and people work together, symbiotically, to perform useful work under circumstances (physical, perceptual, and cognitive) in which neither could do the job alone. Not surprisingly, the major impediment to making this a reality lies in the difficulty encountered in supervising and teaching semiautonomous machines.

This entry explores the background, issues, and methodology associated with AI prostheses. The discussion is illustrated with four examples; an omnidirectional wheelchair that has the potential to find its way around the house; a reading aid prosthesis that tells the user what he/she sees; a manipulation aid that can feel objects and manipulate them according to the task at hand; and a visual-language-communication aid that supports "talking" when normal expressive and receptive language function has been lost.

**Human-Machine Symbiosis.** People with physical, sensory, and cognitive disabilities have long sought to benefit from prostheses and orthoses. Although the distinction will not be emphasized, it should be known that traditional prostheses are devices that replace anatomy and function (an artificial foot for an amputee) whereas orthoses are devices that augment existing anatomy and function (a back brace or eye

glasses). AI prostheses, the concern of this entry, augment function and have little if any direct impact on anatomy. They replace defective body functions with "intelligent" machine functions. They go beyond algorithmic programming and include symbolic reasoning.

False teeth, artificial hips, eye glasses, and hearing aids are familiar assistive devices. Books, film, and computer programs are, in the limit, tools of civilization that augment the limits of one's memory and support communication across time, distance, and culture. From this point of view, the history of technology may be seen as a succession of efforts to extend basic human capacities for transportation, manipulation, perception, and reasoning. Everyone is handicapped by the discrepancy between his/her imagination and the physiological machinery.

Most disabled individuals are strongly motivated to be whole and to live normal lives. They seek the same independence of action, perception, and thought that most people enjoy, however briefly. For a disabled individual the prosthesis is a tool for self-fulfillment and independence (1).

Rapid evolution of machine intelligence is of profound importance to physically limited people, for it makes several new classes of prostheses possible. These include smart mobility aids, pattern-recognizing sensory aids, semiautonomous manipulation aids, and cognitive aids. In all cases impairment results in reduced information-handling capacity. Accordingly, the performance of any given task will be more demanding when a prosthesis is used than when it is performed by an able-bodied individual. This is the driving force to evolve "intelligent prostheses."

For over 20 years it has been possible to build prosthetic devices that can move, sense, store, and retrieve data. Recently, prostheses have been capable of many kinds of signal processing. But until now it has not been possible for prostheses to interpret signals or plan movement. This limitation is very important, for there is a profound "mental work load" difference between perception and control of one's own body and conscious control of an external machine (prosthesis). Now, AI can assume some of this signal-understanding and task-planning work load.

Machines that have access to human supervisory command and control can be expected to perform more robustly and across a wider range of operating circumstances than fully automatic devices. Although this assertion is offered without proof, there is considerable evidence of its validity (2). Most hazardous materials-handling tasks are performed by human-supervised and -controlled remote manipulators. Underwater exploration, construction, and servicing is done exclusively by telerobots. NASA's plans for space-station construction and servicing call for telerobots until such a time that autonomous robots have the required capability and flexibility (3). For the disabled individual the machine can, with human assistance, be made to adapt to uncertain circumstances, can be directed toward time-varying goals, and can perform complex pattern-recognition tasks that are well beyond the current bounds of machine perception and planning.

**Historical Context for AI Prostheses.** Prostheses may be divided into four functional categories. Mobility aids (leg substitutes) are among the first recorded examples of prosthetic devices (e.g., peg legs). In the form of wheelchairs, they have come to symbolize disability. Sensory aids (typically vision and hearing-augmentation devices) are perhaps the most common orthotic devices and include a wide variety of eye glasses

and hearing aids. The distinction between able and disabled users is not always obvious, for the vast majority of eye glasses (sun glasses) and hearing devices (speakers) are for the able bodied. Manipulation aids (arm and hand substitutes) are traditionally associated with amputation but are increasingly seen as replacing lost manipulative function rather than lost anatomy. Most recently, a new class of assistive devices, cognitive aids, begins to hold some promise for functional replacement of defective mental (communicative and intellectual) functions.

Mention should also be made of an emerging class of assistive technologies known as functional electrical stimulation (FES). The FES approach to functional restoration is based on direct electrical stimulation of residual muscle and sensory organs. Unlike orthoses and prostheses, there is direct, invasive control of the user's neuromuscular system. Although auspiciously the closest thing to true anatomical restoration, the success of FES depends heavily on the state of the art in neuromuscular physiology (4). When this knowledge base is deep enough to proceed, FES may then take advantage of developments in the field of AI prostheses.

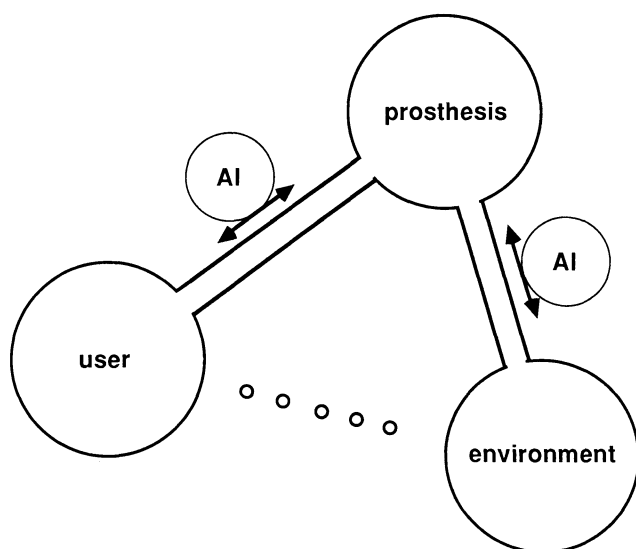
#### Theoretical Context and the Need for AI Prostheses

**User-Machine-Environment Triad.** It is suggested that the ideal prosthesis will be an elegant balance of human and machine functions that act symbiotically to achieve human goals. As represented in Figure 1, the application of AI methods can be expected to have a major impact on the human-to-machine communication link and the machine-to-environment interaction interface. Use of a desk-top word processor serves as a limited example of the user-machine-environment triad. The user communicates his intentions (however imperfectly) to the word-processing computer program; some classes of input error are corrected by the machine, others by additional user

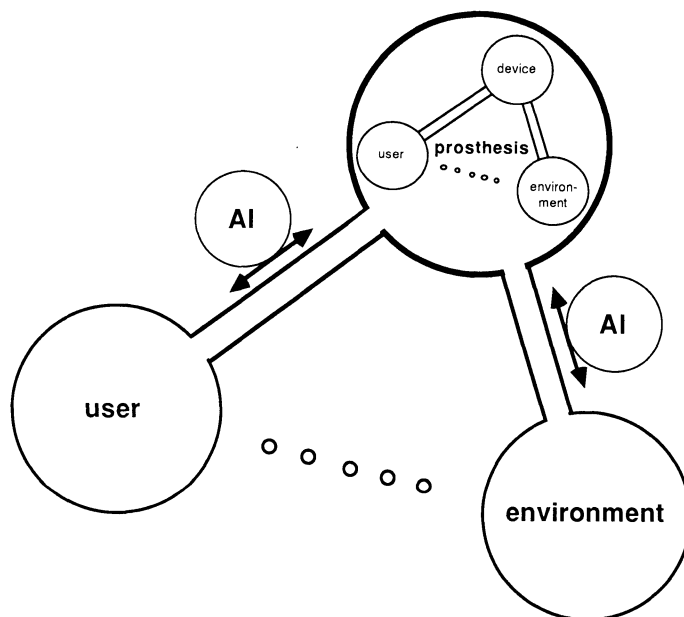
input; almost all printing tedium is managed by the computer. The exact division of labor between the prosthesis and the user changes dynamically, according to the nature of the task, experience gained in doing this class of tasks, capabilities of the prosthesis, and personal preference.

The overall context for AI in prostheses is represented in Figure 2, where the concept of a user-machine-environment triad is extended to acknowledge that the full power of this partnership will not be attained until the machine's knowledge base contains a verifiable "world model," a model that is, in part, devoted to representing the user in his setting and to managing the dialogue between man and machine. Whereas most approaches to machine vision rely on computational techniques to identify and extract features of the world from raw data, a strategy based on symbiosis assumes that a human operator will identify and label critical features for machine representation and tracking. This is particularly relevant when the utility of the system will be measured in terms of its performance as a prosthesis.

**Mobility Prostheses.** In the context of mobility aids, the primary demands for machine intelligence are associated with path-planning navigation, obstacle avoidance, and real-time control (see Path planning and obstacle avoidance). The objective is to make externally powered mobility available to the widest possible range of drivers under diverse operating conditions. This includes people who might otherwise have difficulty controlling the vehicle (motor or sensory disability) or difficulty recalling and planning their movement (memory or cognitive disability). Although path planning is an active area of fundamental research (5), the most severe constraint on AI applications to mobility prostheses lies in the requirement that sensing and planning operations must take place in real time and use fault-tolerant computers compatible with wheelchair vibration, power, and environmental effects.



**Figure 1.** User-prosthesis-environment triad: For the disabled individual who is unable to access or respond to the environment on a par with others, a prosthesis becomes the critical bridge to normal independent living. AI promises to facilitate communication between the user and the prosthesis and to improve the performance of the prosthesis as a tool for manipulating the environment. An intelligent prosthesis will be used if its actions can easily be programmed and supervised.



**Figure 2.** Intelligent prosthesis extended to include an internal world model: A next generation of intelligent prosthesis should include internal representations of the world, including a model of itself, its environment, and its user. The acquisition of this "knowledge" can and should be user-interactive.

**Sensory Prostheses.** In the domain of sensory prostheses, sensory "substitution" has proven to be a very elusive goal. Whenever signals normally processed by one sensory channel (e.g., eyes for vision) are mapped to another channel (e.g., tactile displays), one encounters a mismatch that may lead to an acute information-processing overload, fatigue, and limited acceptance of the prosthesis (6). The central task for AI in sensory prostheses is one of signal understanding and pattern recognition. It is not enough that each pixel in a video-camera raster-scan display be forwarded to a vibrotactile skin "display." Rather, key features of the world must be extracted from machine sensors and forwarded to the user at an appropriate rate. For example, a  $100 \times 100$  element tactile display may be much harder to use than an aural declaration that "there is a head-high obstacle three feet in front of you."

**Manipulation Aids.** Manipulation aids incorporating robotics and AI technology have only recently become feasible. Previously, only skilled operators (e.g., pilots) have been able to control more than two degrees of freedom (a typical airplane has three degrees of freedom during standard maneuvers). The functional analog of a human upper limb, a robotic arm, must have at least six degrees of freedom (plus grasp) to hold an object in an arbitrary orientation. Although it has long been possible to build externally powered "artificial arms," it has not been possible to control them (7). Key contributions from AI include the development of motion programming languages such as AL, VAL, and XPROBE (8–10). Important active areas for research and development include manipulator path planning, spatial reasoning, temporal reasoning, machine vision, and tactile sensing.

**Cognitive Prostheses.** The newest category of prostheses, cognitive aids, are only now becoming feasible with the maturation of AI tools for natural-language processing (qv) and non-numeric knowledge representation (qv). Correlates of this development have been the evolution of highly visual programming environments (qv) (e.g., windows, high-resolution bit-mapped displays, and iconic representation) and programming languages (e.g., LISP, PROLOG, Smalltalk) that facilitate symbolic computation. Note that these tools were developed to help the "cognitively limited" AI researcher deal with the planning and performance assessment of complex programs, a task akin to the routine struggle of a cognitively disabled individual to deal with the planning and operation of his/her own communication. Technical advances can now be applied to expedite the use of BLISS and other symbol systems developed for people with communicative disorders. Although proven to be effective, their utility on "flash cards" and "communication boards" has always been too limited for widespread adoption (11). Advances in graphical and temporal reasoning will enhance the potential of these symbol systems as cognitive prostheses.

### Specific Properties and Design Constraints for AI Prostheses

**User Expertise and Task Structure.** Intelligent prostheses will be sophisticated machines. Physically disabled people who need these devices will not (in general) be comparably sophisticated or technically literate. Accordingly, user training should be built into the assistive device. Product usage should be "self-explanatory." The sense of this may be found in some computer-application programs that include "context-sensi-

tive on-line help." One may anticipate that an intelligent prosthesis will be expected to function well in unstructured task situations about which the prosthesis may itself need to learn. In this context the user must help the prosthesis. Such an expression of reciprocity in the human-machine interface represents a novel development and a further expression of the need for dialogue management and a world model that represents the user as well as the environment. The intelligent prosthesis must be interactive with its user in ways and to degrees that are largely without precedent.

**Human-Machine Communication Management.** It is likely that a user/prosthesis dialogue will be rather more like an interpersonal conversation than a "programming" session. Characteristically, it will be implemented with the spoken word rather than the written word. Wherever possible, graphic I/O will be employed. Machine recognition of connected speech will be required, and machine synthesis of speech with intonation and "emotional" coloration will be desirable. When compared to the industrial application of comparable technology, the intelligent prosthesis will be used in relatively unstructured domestic and institutional environments. In a further departure from industrial applications of AI, the prosthesis user will rarely do anything repetitiously. Fortunately, many personal tasks are variations on a theme and may be performed as specific cases of a generic solution. However, the constantly shifting objectives of the user demand that interaction be facile and that the user be relied on to help the prosthesis adapt to changing goals and conditions. This leads to a dynamic allocation of responsibility between man and machine. There might be very little machine-initiated activity during training for a new task and then a gradual shift toward largely autonomous task performance by the machine as the user gains confidence in the prosthesis.

**Self-Expression and Aesthetics.** It is not enough to create intelligent prostheses that only address maintenance functions, for once satisfied, the human spirit seeks further expression in recreation and the arts. There is good reason, and some evidence, to suggest that intelligent prostheses can support the disabled individual's self-expression in dance, music, graphics, and performing arts. The Stanford Robotic Aid (discussed in detail below) has been used by disabled individuals to express themselves in the form of "dances" choreographed for performance by the manipulator or by combinations of robots and human performers (12,13).

### Analytic Methods for Intelligent Prostheses

There is a relative paucity of analytic methods for the assessment of any prosthesis, let alone an intelligent prosthesis. This is not to say that scientific methods are not applicable but that there is very little experience or literature on the subject at this time. Research in three areas of human-machine study have proven useful. Supervisory control of vehicles and manufacturing processes bears a close resemblance to user interaction with an intelligent mobility or manipulation aid. The results from decision-support-system modeling and analysis (6) are relevant to the design and assessment of sensory and cognitive prostheses. History lists and audit trail methods evolving in the AI development environment should be important tools for the assessment of human-prosthesis interaction patterns and performance measurement.

### Intelligent Prostheses Applications

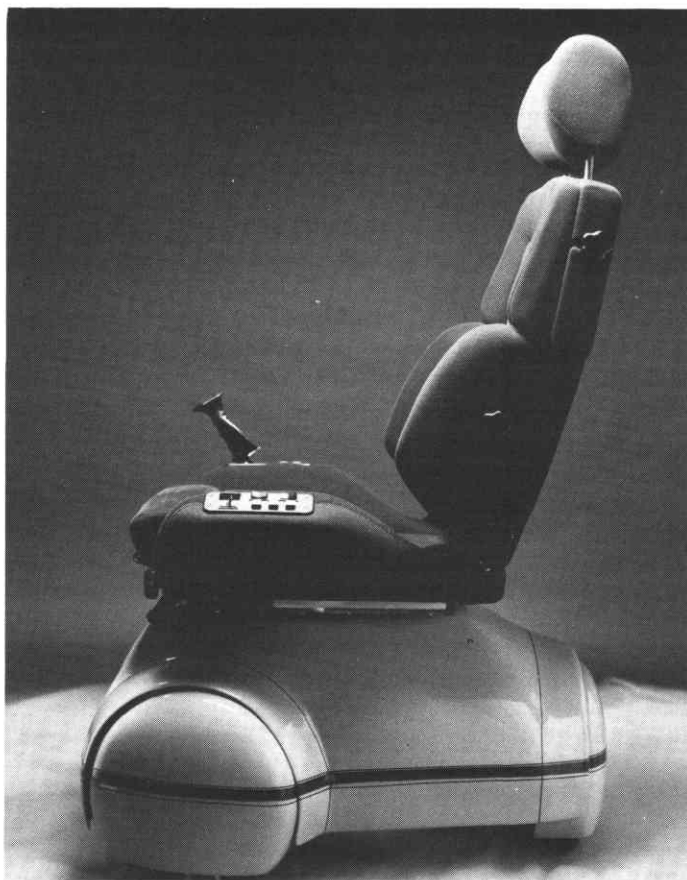
**Intelligent Omnidirectional Mobility.** In October 1982 a team of four master's degree candidates in the Department of Mechanical Engineering at Stanford University were challenged by the Veterans Administration (VA) to create a "new wheelchair." Their specifications were limited to the statement that the vehicle should have the same degree of mobility on level ground that ambulatory people have and that the vehicle should not have a "footprint" larger than that of an ambulatory person. The objective was to provide wheeled mobility in the domestic and vocational spaces built for ambulatory persons. More formally, the vehicle was to have three independent degrees of freedom (forward/backward, left/right, and rotation) equivalent to walking persons. Beyond mobility, it was suggested that an "intelligent" wheelchair would be able to adjust to the driving skills of its user. Factors affecting driver skill include reaction time, strength, range of motion, postural stability, spasticity, tremor, and visual perception. The wheelchair-operating system should recognize high-risk driving conditions and self-limit its speed and/or rate of directional change. In addition to adaptive and safety features, onboard computation was expected to perform the functions of an automatic pilot for those who needed mobility but could not drive. Several of these adaptive features were demonstrated in a proof-of-concept "smart wheelchair" (14).

The design team and their advisors were largely successful. Now, following 3 years of refinement and production engineering, the Alexis "personal vehicle" (Fig. 3) will be the first of a series of mobility aids that incorporate computers and adaptive programming. Although not yet intelligent in the sense of long-range AI objectives, this vehicle lays the foundation for "intelligent wheelchairs." The onboard computer now manages power resources, coordinates motor speed to provide closed-loop directional control, adapts to user skill levels, and accommodates collision-avoidance sensors. Continuing development builds on this capability to provide rudimentary path planning, navigation, and collision avoidance.

**Reading for People with Severe Visual Impairment.** The earliest and most widely used approach to reading aids for the blind is itself an "artificial language" called braille. This  $2 \times 3$  raised-dot cell matrix is typically imprinted on paper and read by touch. Grade I braille codes correspond to English (or other European language) characters and can be machine transcribed. Grade II (moderately contracted) braille can be partially transcribed by machine. Grade III (highly contracted) braille is too idiosyncratic for algorithmic transcription but may be subject to an expert-system (qv) approach. Early braille texts were produced by hand. Now, there are several computer-controlled braille embossers available.

Although reading for partially sighted individuals is an important concern, it is not reviewed here because the need for machine intelligence is either absent or converges with the needs of nonsighted individuals.

In the mid-1970s several groups assembled devices to assist blind individuals with the task of reading printed material directly, without translation to braille. Successful work evolved along two strategic paths. In one case, represented by the Kurzweil reading machine (Fig. 4), the emphasis was on machine recognition of the printed word and machine-synthesized voice output. Through automated page scanning and optical character recognition (qv), these devices free the user



**Figure 3.** Alexis, omnidirectional motion and smart-wheelchair precursor: This omnidirectional personal vehicle can move forward and backward, side to side, and turn about any center of rotation (including its own geometric center). It is designed to move with the same agility that ambulatory individuals have in flat interior spaces (especially domestic spaces). Motion mechanics are based on a patent by William La. A proof-of-concept model was developed by the Veterans Administration Rehabilitation Research and Development Center in Palo Alto, CA. That prototype was the first mobility prosthesis (smart wheelchair) to include an onboard microcomputer and adaptive control features. A descendant of the Alexis is to be marketed by Intex Inc.

from the mechanics of braille reading, leaving the user free to concentrate on comprehension. The text-to-braille transcription stage is bypassed, and virtually all machine-compatible material becomes accessible. Over the past 10 years these machines have evolved to handle a variety of print fonts, a diversity of print sizes, and a quality of voice synthesis that allows long-term listening. The system has remained expensive and bulky, such that most usage is institutional. Though these devices now have only rudimentary machine intelligence, there is considerable potential for improved page-layout handling and symbolic programming to augment optical character-recognition algorithms. Steady improvement appears likely, and one now finds related devices available to the able-bodied for conversion of printed text to machine-readable text suitable for editing, electronic transmission, and speech output.

In a collateral line of development, the Optacon (Fig. 5) was developed with an emphasis on portability and reliance on human learning. In this system the user holds the optical sensing element and hand scans the page while simultaneously





**Figure 4.** Fixed-base reading machine: This table-top page scanner uses optical character-recognition algorithms to read a variety of print fonts and convert them to machine-readable codes suitable for text processing, speech synthesis, and braille embossing. Developed over a period of approximately 10 years, this reading machine is now available in one or more models from Kurzweil-Xerox Inc. It is widely available in libraries.

"reading" an array of vibrotactile stimulators. An optical sensing array is mapped to the vibrotactile display. Successful use of the Optacon depends heavily on user aptitude, adaptation, motivation, and training. However, portability is an important feature and has contributed significantly to making Optacon the most widely owned and used "high-technology" prosthesis available today. Although the Optacon is not AI oriented, a follow-up project at the VA Rehabilitation Research and Development Center in Palo Alto, California, does take advantage of some AI methodology to relax the constraints on accurate manual tracking, to extend the range of font styles and sizes, and to include synthesized speech output (15). With the introduction of custom integrated circuits to perform low-level image-processing tasks, the system unit becomes available for more extensive AI application.

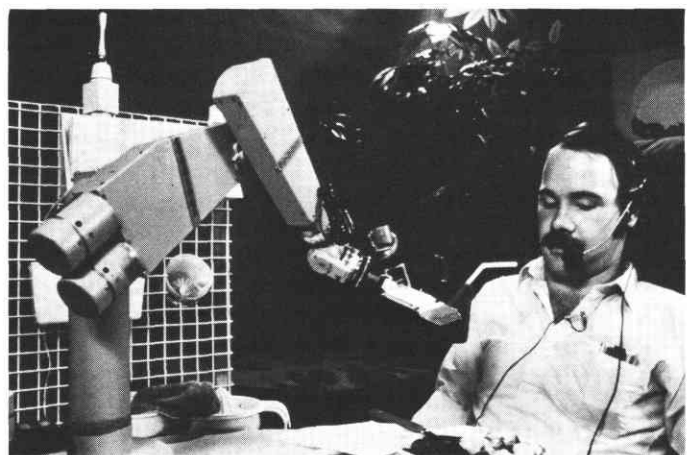


**Figure 5.** Portable reading machine: The Telesensory Systems Optacon weighs approximately 5 pounds and uses  $6 \times 12$  element vibrotactile index-finger display to let the user "see" printed characters. Although considerable training is required to read with this system, flexibility and portability have made it one of the most widely used "high-technology" sensory aids.

Recently, Collins and Deering (16) have built a "proof-of-concept" real-time scene analyzer for the blind pedestrian. They attempted to extract features of the environment that are particularly relevant to safe mobility and navigation. These features included curbs, poles, fences, and moving objects (e.g., cars). Range and scale information are presented to the user by a one-dimensional tactile array, and supplemental information is announced by synthesized speech. The concept has been demonstrated, but a great deal more needs to be done with the human interface and recognition technology.

**Development of a Robotic Manipulation Aid.** Prostheses for the replacement of lost arm and/or hand anatomy have been built since the thirteenth century. Although most of these were the upper-limb equivalent of a peg-leg, a few devices were jointed and capable of holding a variety of tools. The earliest attempts to use digital computer technology were directed toward control of multiple degree of freedom upper-limb orthoses (17,18). This type of orthosis is typically an exoskeletal device that replaces lost muscle function (as in poliomyelitis) while taking advantage of the user's own sense of touch. However, the full expression of computer technology and an opportunity to introduce AI coincided with the realization that the goal of a manipulation aid was properly that of replacing lost function, not lost anatomy (19-21).

The Stanford University Robotic Aid Project is the most recent project in a series of efforts to bring "machine intelligence" to the task of replacing lost or deficient manipulative function. The feasibility of using industrial robotics (qv) technology has been demonstrated through the development of two prototypic robotic aids, one for clinical evaluation and one for technical evaluation (22-24). Each is composed of an electromechanical manipulator with six degrees of freedom (Unimation PUMA-260) that can be controlled by spoken commands, joystick displacements, stored programs, and rudimentary robotic hand sensors (Fig. 6). Although over 25 quadriple-



**Figure 6.** First-generation Stanford robotic aid: This is the first manipulation aid to use a human-scale industrial manipulator to replace upper-limb function for persons with quadriplegia and other severe motor impairments. The user can program and control the manipulator by a combination of spoken commands and analog inputs. Approximately 100 people have been trained to use the system. As a successful demonstration of feasibility, the findings of this study period led to detailed specifications for a second-generation aid intended to demonstrate the utility of robotic aids (VA sponsored).

**Table 1. Hypothesized Requirements for a Utilitarian Robotic Aid**

Objective assessment of utility is possible
IF the robotic aid is instrumented to record a time history of user/robot transactions; and
IF quantitative clinical studies are performed to assess the contribution of each major robotic aid feature;
Utility can be enhanced
IF user/robot transactions can be made as brief and as natural as possible
Natural human-machine interaction is possible
IF a "dialogue manager" acts to mediate user/robot interaction such that an effective dialogue is maintained given imperfect speech processing; and,
IF "internal state" information about the robotic aid is conveyed to the user by voiced and graphic displays; and
IF the degree of focused attention required to supervise the robot is reduced through autonomous machine execution of many real-time control tasks.
Autonomous machine functions are feasible
IF the robotic aid has an internal representation of its own status and surrounding environment; and
IF command, control, and sensor data can be integrated with model reference data; and
IF sufficient computational power is available for data-driven motion planning.
Data-driven control is realizable
IF the environment can be sensed and processed in real time; and
IF user command and control inputs can be processed in real time.
User and environmental safety are assured
IF the user is outside the manipulator's working volume; and
IF the user can redirect ongoing action within 250 ms; and
IF an independent computer acts on sensor data to constrain manipulator speed, force, and working volume.

gic individuals have been trained to use the clinical systems and have gained some degree of direct control of their physical surroundings, it is clear that the robotic aid, in its current form and present capabilities, is not ready for widespread distribution. Experience with the "first-generation" robotic aid helped define key areas where further research and development are needed.

Key requirements for a second-generation robotic aid (now under development) were based on experience with the first-generation aid. Quoting from the project planning document, Table 1 defines the set of nested hypotheses that must hold if robotic manipulation aids are to succeed.

The testing of these hypotheses may be undertaken as a structured implementation of robotic-aid capabilities required to perform daily living activities (Table 2). Briefly, these capabilities are defined as follows.

**Commands.** Discrete transactions between the user and the robot, finite in context and duration, commands allow the operator to change operating conditions.

**Controls.** Continuous temporal interaction between the user and the robot allows the operator to pilot the arm and drive mobile base.

**Dialog Management.** A set of procedures and rules designed to assure that effective two-way communication is maintained between the operator and the machine despite noisy command and control channels.

**Autonomous Planning.** It is performed by the machine when sensed data are used by task-specific programs to make navigating decisions without requiring human intervention (all robotic aid actions are subject to human supervision and veto).

**Grasp Automation.** The property of a robot that allows an object to be picked up and held appropriately without step-by-step human interaction.

**Table 2. Manipulation-Aid Features Required for Daily Living Tasks.<sup>a</sup>**

Tasks	Functional Capabilities							
	Command	Control	Dialogue	Planning	Grasp	Manipulation	Reflexes	Mobility
Maintenance								
Food preparation	■	■	□	□	■	■	□	■
Food service	■	■	□	□	■	■	□	□
Personal hygiene	□	■	□	□	■	□	■	□
Personal grooming	■	■	□	■	■	■	■	□
Clothing management	□	□	□	■	■	■	□	■
Appliance usage	■	■	□	■	■	■	□	■
Vocation								
Storage and retrieval	■	□	□	□	■	■	□	■
Equipment operation	■	□	□	□	■	■	■	■
Assembly	■	■	■	■	■	■	□	□
Word processing	■	□			□	□		
Computing	■		□					
Materials processing	■	□	□	□	■	□	■	■
Recreation								
Reading	■	□		□	□	□	□	
Film and video	■				■	■	□	■
Performing arts	□	■	□	□	■	■	□	■
Graphic arts	□	■	□	□	■	■	□	
Sports	□	■	□	■	■	■	■	■
Social interaction	□	□	■		□	□	□	■

<sup>a</sup> In this tabulation specific technical features of a robotic aid are associated with the daily living tasks that require those features. The objective of the exercise is to determine the nominal specifications for a robotic aid that would be demonstrably utilitarian. Symbols: (blank), not needed; □, should have; ■, must have.

**Manipulation.** The ability to move objects from one place to another while maintaining a desired orientation of the hand and avoiding objects.

**Programmable Reflexes.** They protect the operator and the robot from some kinds of adversity. They are quick acting and are not subject to real-time human interaction.

**Mobility.** Robot mobility vastly extends the manipulation aid's working volume beyond that of a desk-top work station.

The second-generation Stanford Robotic Aid (Fig. 7) has been assembled from the following subsystems and is about to enter clinical testing.

**Command Input.** A speaker-dependent voice-recognition computer is used for command entry (a keyboard is also available).

**Command Output.** A speech-synthesis computer is used for dynamic command verification. Operator status information is presented on an alpha-numeric display.

**Control Inputs.** A two-degree-of-freedom ultrasonic head control unit is used by severely impaired operators to pilot the arm and drive the mobile base (a six-degree-of-freedom joystick provides extended control for less severely impaired persons).

**Dialog Management.** A personal computer (programmed in in Pascal) manages command, control, and graphic-display interactions at the user interface.

**Autonomous Planning.** A computer programmed in MicroPower Pascal is used to perform sensor integration, path planning, and coordinate transformations.

**Grasp Automation.** Force-and-optical-proximity-driven grasp automation is controlled by a dedicated microcomputer programmed in MicroPower Pascal.

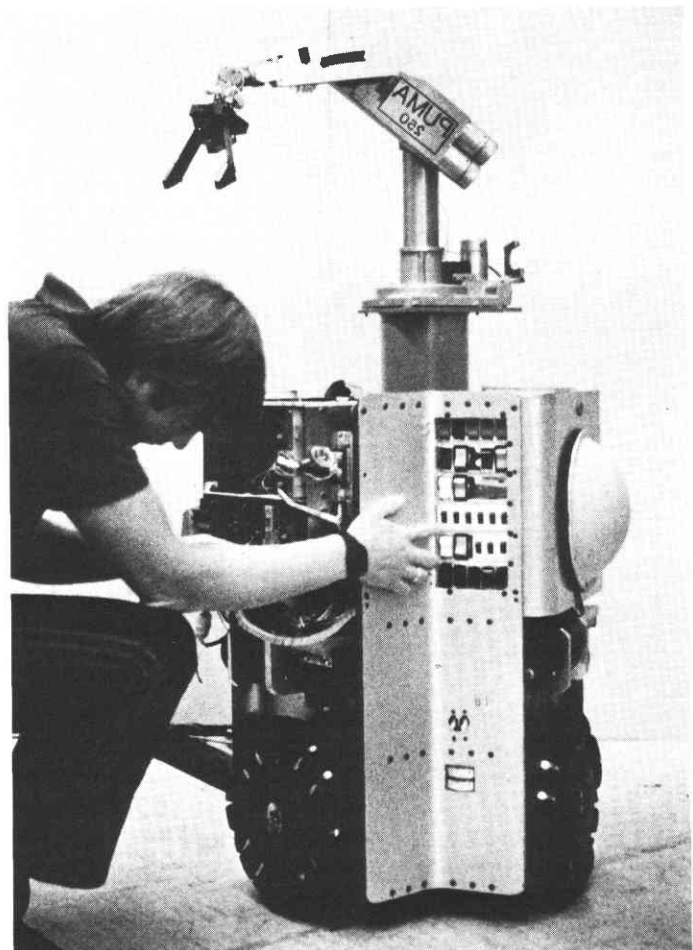
**Manipulation.** An industry standard manipulator is controlled by dedicated 6502 microprocessors. High-level motion commands are generated by the personal computer listed in *Dialog Management*.

**Programmable Reflexes.** Reflexes are implemented in the grasp controller and motion planner.

**Mobility.** The mobile base has dedicated wheel-motion controllers equivalent to the arm-joint controllers.

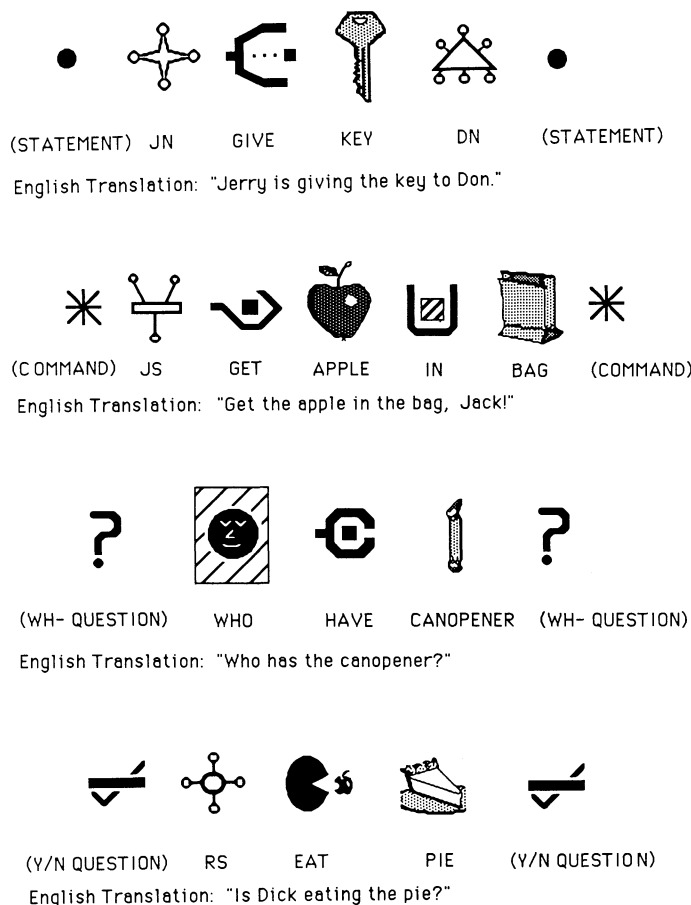
**Visual Communication Aids for People with Language Impairment.** Developments in AI, computer graphics, and human-computer interface design are making possible new ways of treating aphasic individuals. Aphasia is the loss of natural-language abilities such as speaking, understanding, reading, and/or writing. It occurs most frequently because of stroke or traumatic head injury. Although some of these people do recover partly or fully, large numbers never do. The communication needs of these chronically aphasic individuals are poorly met by traditional therapies or current assistive devices.

Specifically, a computer-based visual communication aid for severely impaired aphasics is under development by Steele (25) and Lakin (26). Notable features of their device include a "mouse" (pointing device) that is effectively operated by one hand (many persons with stroke and head injury have unilateral paralysis); a high-quality visual icon-based graphic communication system; graphic access and feedback schemes for improved user performance; translation facilities, text dis-



**Figure 7.** Second-generation Stanford robotic aid: This advanced robotic aid includes the following key elements: command input: a Kurzweil (model KVS 1.1) speaker-dependent voice-recognition computer is used for command entry; command output: a DECTALK speech-synthesis computer is used for dynamic command verification. Operator status information is displayed alpha-numerically; control inputs: a two-degree-of-freedom ultrasonic head control unit is used by severely impaired operators to pilot the arm and drive the mobile base; dialog management: an IBM PC/AT personal computer manages command, control, and graphic-display interactions at the user interface (programmed in Pascal); autonomous planning: a DEC LSI 11/73 (soon to be microVAX-II) programmed in MicroPower Pascal is used to perform sensor integration, path planning, and coordinate transformations; grasp automation: force and optical proximity driven grasp automation is controlled by a dedicated DEC KXT-11 microcomputer programmed in MicroPower Pascal; manipulation: a Unimation PUMA-260 manipulator is controlled by dedicated 6502 microprocessors receiving high-level motion commands from the LSI 11/73; programmable reflexes: are implemented in the grasp controller and motion planner; mobility: the mobile base has dedicated wheel motion controllers equivalent to the arm-joint servos; motion planning is performed by the LSI 11/73 while safety features such as the instrumented bumper are managed by DEC KXT-11s.

plays, and voice output for communication with those unfamiliar with the system (e.g., family and friends); and implementation on a transportable personal computer system (Apple Macintosh-512). The approach takes advantage of an aphasic individual's residual strengths and abilities. Aphasia usually results from damage to the left hemisphere of the brain, which tends to handle language processing. Impairment often in-



**Figure 8.** Visual communication aid for severely impaired aphasics: An example of the bit-mapped user icons created by severely aphasic individuals to implement a visual communication language.

cludes right hemiparesis (paralysis of the right arm and/or leg), which leaves only the left hand fully functional. In stressing use of the unimpaired visual-processing capabilities of the user's right hemisphere, this approach circumvents lost natural-language abilities. The work is based on earlier efforts using index cards (27) which demonstrated the effectiveness of iconic symbols but was ill-suited to becoming a practical communication aid.

A typical interaction script is represented in Figure 8. Preliminary studies with computer-mediated visual-communication languages have been very promising, with users showing both improved performance and enthusiasm for the approach.

In related work Levin et al. (28) have reported effective guidance of poststroke memory-impaired individuals in the preparation of meals (inability to prepare meals may result in institutionalization of persons living alone). Their work supports the use of task visualization and some instructional narrative (these users are not aphasic). Results indicate that an interactive "cookbook" can demonstrably help overcome short-term-memory impairment in the context of well-structured tasks.

### Summary

This has been a brief review of the state of the art and the potential for AI applications to mobility, sensory, manipula-

tion, and cognitive prostheses. In all cases the work is at an early stage of development, and progress will depend heavily on the understanding of both "natural" and "artificial" intelligence. Readers are invited to seriously consider personal investment of their expertise in the development of intelligent prostheses.

### BIBLIOGRAPHY

1. L. Leifer, High Technology, by Design, for the Handicapped, in V. W. Stern and M. R. Redden (eds.), *Technology for Independent Living*, American Association for the Advancement of Science, Washington, DC, pp. 143-157, 1982.
2. M. W. Thring, *Robots and Telechairs: Manipulators with Memory, Remote Manipulators, and Machine Limbs for the Handicapped*, Wiley, New York, pp. 28-29, 1983.
3. Reference 2, pp. 245, 256, and 259.
4. L. Vodovnik, T. Bajd, A. Kralj, F. Gracanin, and P. Strojnik, Functional Electrical Stimulation for Control of Locomotor Systems, *CRC Critical Reviews in Bioengineering*, CRC, Boca Raton, FL, pp. 63-131, 1981.
5. T. Lozano-Perez, Automated Planning of Manipulator Transfer Movements, *IEEE Trans. Sys. Man Cybernet.* **SMC-11**(10), 681-689 (1981).
6. T. B. Sheridan and W. R. Ferrell, *Man-Machine Systems: Information, Control, and Decision Models of Human Performance*, MIT Press, Cambridge, MA, pp. 61-90, 1974.
7. D. E. Whitney, The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators, in M. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M. T. Mason (eds.), *Robot Motion*, MIT Press, Cambridge, MA, pp. 287-304, 1982.
8. R. Finkel, *AL: a Programming System for Automation*, AIM-243, Stanford University Artificial Intelligence Laboratory, Stanford, CA, 1974.
9. B. Shimano, *VAL—A Robot Programming and Control System*, Unimation, Danbury, CT, 1979.
10. P. D. Summers and D. D. Grossman, "XPROBE: An experimental system for programming robots by example," *Int. J. Robot. Res.* **3**(1), 25-39 (1984).
11. H. Gardner, E. B. Zurif, T. Baker, and E. Baker, "Visual communication in aphasia," *Neuropsychologia* **14**, 275-292 (1976).
12. G. Curtis, Intentional Machine Choreography: Exploring the Expressive Potential of the Robot, *VA Rehabilitation Research and Development Center Report'84*, Palo Alto, CA, 1984.
13. M. K. Apostolos, Exploring User Acceptance of a Robotic Arm: A Multidisciplinary Case Study, Ph.D. Dissertation, Stanford University, School of Education, Stanford, CA, 1984.
14. W. H. T. La, T. A. Koogle, D. L. Jaffe, and L. Leifer, Towards Total Mobility: an Omnidirectional Wheelchair, *Proceedings of the Rehabilitation Engineering Society of North-America*, Washington, DC, pp. 75-77, 1981.
15. R. D. Steele, D. Hennies, R. Smith, R. Miranda, and D. Ponta, A Microprocessor-Driven Optacon Interface to a Reading Aid for the Blind, *Proceedings of the RESNA/ICRE Joint Conference*, Ottawa, ON, June 1984, pp. 585-586.
16. C. C. Collins and M. F. Deering, A Microcomputer Based Blind Mobility Aid, *IEEE Conference Proceedings: Frontiers of Engineering and Computing in Health Care*, Los Angeles, CA, 1984, pp. 212-216.
17. J. Reswick and K. Mergler, Medical Engineering and Progress Report on the Case Research Arm Aid, Report EDC 4-64-4, Engineering Design Center, Case Institute of Technology, 1964.

18. M. Moe and J. Schwartz, Coordinated, Proportional Motion Controller for an Upper-Extremity Orthotic Device, *Proceedings of the Third International Symposium on External Control of Human Extremities*, Dubrovnik, Yugoslavia, 1969, pp. 295–305.
19. E. Heer, G. A. Wiker, and A. Karchak, Voice Controlled Adaptive Manipulator and Mobility System for the Severely Handicapped, *Proceedings of the Second Conference on Remotely Manned Systems Technology and Applications*, June 1975, pp. 85–86.
20. J. Vertut, J. Charles, P. Coiffet, and M. Petit, Advance of the New MA23 Force Reflecting Manipulator System, *Second CISM-IFTOMM Symposium on Theory and Practice of Robots and Manipulators*, Warsaw, Poland, 1976, pp. 307–322.
21. W. Schneider, G. Schmeisser, and W. Seamone, "A computer-aided robotic arm/worktable system for the high-level quadriplegic," *IEEE Comput.* 41–47 (January 1981).
22. L. Leifer, Restoration of Motor Function—a Robotics Approach, in J. Raviv (ed.), *Uses of Computers in Aiding the Disabled*, North-Holland, New York, pp. 3–17, 1982.
23. L. Leifer, Interactive Robotic Manipulation for the Disabled, *Proceedings of the Twenty-Sixth IEEE Computer Society International Conference (COMPCON)*, San Francisco, CA, March 1983, pp. 46–49.
24. R. W. Wicke, H. F. M. Van der Loos, R. Awad, K. G. Engelhardt, and L. Leifer, Evolution of a Vocabulary for Voice Control of a Robotic Aid, *Proceedings of the Second International Conference on Rehabilitation Engineering*, Ottawa, ON, June 1984, pp. 121–123.
25. R. Steele, Computer-Aided Visual Communication for Severely Impaired Aphasic, *Rehabilitation Research and Development Center Report'84*, VA Medical Center, Palo Alto, CA, 1984.
26. F. H. Lakin, A Graphical Communication Environment for the Cognitively Disabled, *Proceedings of COMPCON'83, Twenty-Sixth IEEE Computer Society International Conference*, San Francisco, CA, 1983, pp. 39–45.
27. E. H. Barker, T. Berry, H. Gardner, E. B. Zurif, L. Davis, and A. Veroff, "Can linguistic competence be dissociated from natural function?" *Nature* **254**, 509–510 (1975).
28. S. P. Levin and N. L. Kirsch, COGORTH: A Programming Language for Computerized Cognitive Orthoses, *Proceedings of the Rehabilitation Engineering Society of North America Conference*, June 1985, pp. 359–360.

### General References

- S. K. Card, T. P. Moran, and A. Newell, *The Psychology of Human-Computer Interaction*, Erlbaum, Hillsdale, NJ, 1983.
- G. Beni and S. Hackwood, *Recent Advances in Robotics*, Wiley, New York, 1985.
- S. Todd (ed.), *Rehabilitation R&D Progress Reports*, Rehabilitation Research and Development, Office of Technology Transfer (153D), Veterans Administration, New York, 1983–1985.
- R. C. Schank, with P. G. Childers, *The Cognitive Computer (On Language, Learning, and Artificial Intelligence)*, Addison-Wesley, Reading, MA, 1984.
- V. W. Stern and M. R. Redden, *Technology for Independent Living: Proceeding of the 1980 Workshops on Science and Technology for the Handicapped (Stanford, Boston, Houston)*, American Association for the Advancement of Science, Washington, DC, 1982.

L. LEIFER  
Stanford University

The Veterans Administration should be recognized as the primary source of research and development support for the Alexis personal vehicle, the Stanford Robotic Aid, and the Visual Communication Aid.

## PROXIMITY SENSING

The purpose of proximity sensing is to detect environmental information in close proximity to robots. Proximity sensing is studied mostly in robotics (qv) and occupies a unique position in the sensing range between the direct contact of tactile sensing and global imaging of vision (qv) systems. Typically, proximity sensors are mounted on a moving robotic part (e.g., a manipulator) in order to detect the presence or approximate position of nearby objects without actual touch (see Multisensor integration). They could also be used as obstacle detectors to avoid unexpected bumping of objects.

One of the earliest applications employs a magnetic proximity sensor used for machining instead of a limit switch. More sophisticated sensors have been developed that are capable of measuring approximate position and orientation of object surfaces. The sensing range changes depending on the application: less than 1 mm for arc welding, several millimeters for manipulation, a few centimeters for obstacle avoidance of manipulators, and several tens of centimeters for obstacle avoidance of mobile robots (qv).

Proximity sensors for robots are classified in the following categories:

- ultrasonic proximity sensor,
- magnetic proximity sensor, and
- optical proximity sensor.

Ultrasonic proximity sensors measure time of flight of ultrasonic pulses. They are suitable for detecting big objects at relatively long distances. They are often used for obstacle detectors of mobile robots (1) (see Path planning and obstacle avoidance).

The magnetic proximity sensor consists of two kinds of coils: an exciting coil that induces eddy current in a metal object and a detection coil that detects the magnetic change caused by the eddy current. A typical application to welding robots is illustrated in Figure 1. Instead of teaching a complete welding path to a robot, an operator specifies only representative points along the path. The robot controls the position of the torch with respect to two planes using a vertical and a horizontal sensor.

Optical proximity sensors are widely used for manipulators. The principle of the optical proximity sensor is a triangulation with a light projector and a detector. An example of a basic proximity sensor is illustrated in Figure 2a (2). A light beam is projected by a light-emitting diode (LED) through a lens, and the reflected light beam is detected by the detector. It detects the presence of an object in the detection range (the hatched area). The detection range is adjusted by changing the

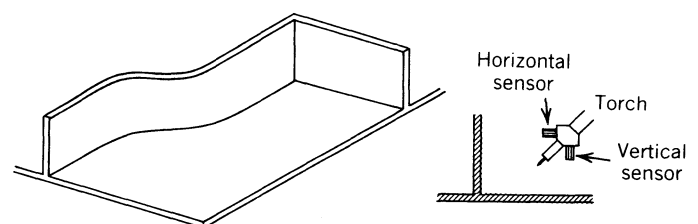
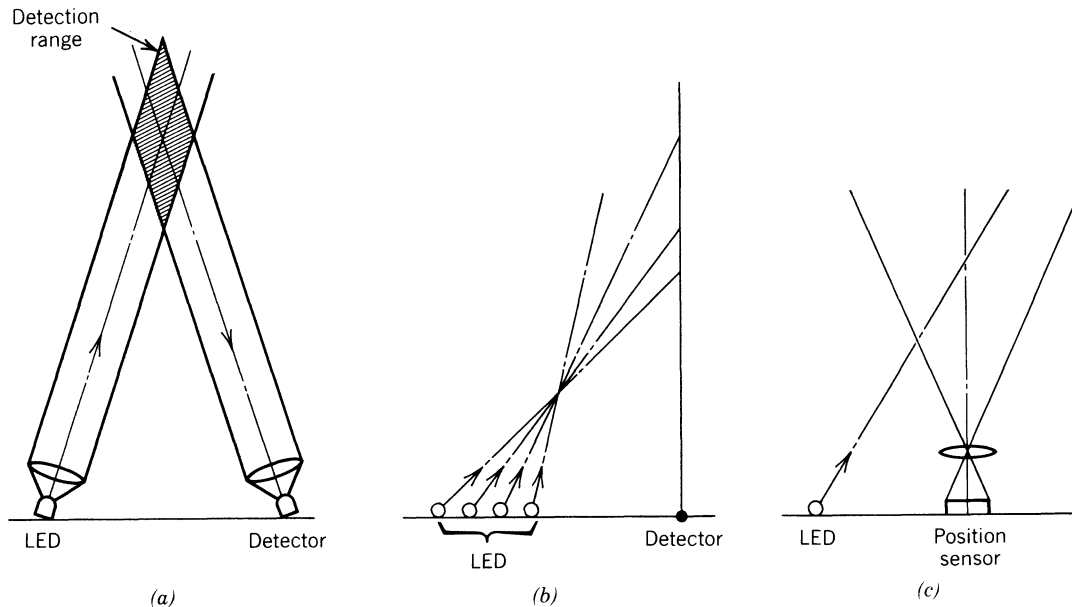
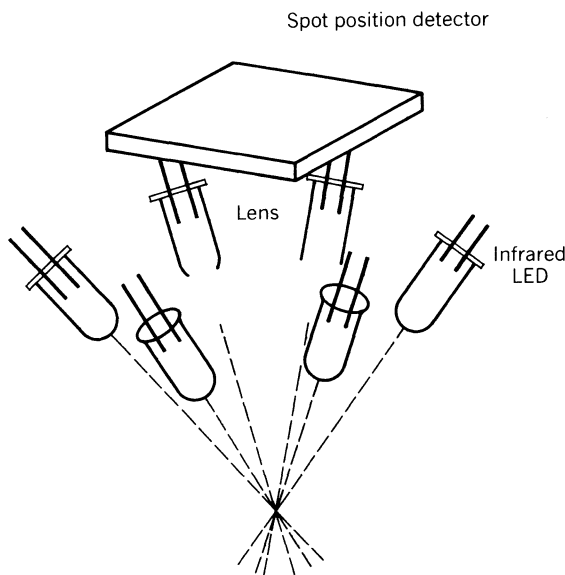


Figure 1. Welding guided by magnetic proximity sensors.



**Figure 2.** Principles of optical proximity sensors: (a) basic configuration; (b) with multiple LEDs; (c) with position sensor.



**Figure 3.** Proximity sensor for measuring surface position and orientation.

configuration of the optical system and the width of the light beam (see Optical flow). Since proximity sensors of this type are simple, they can easily be mounted to robotics parts.

Figure 2b shows an extension of Figure 2a able to detect multiple ranges (3). Multiple LEDs project sharp light beams sequentially, and a reflected beam is similarly detected (lenses and the beam width are omitted in Figure 2). It is easily determined from the timing from which LED the detected light is projected. The range of an object is similarly determined. Note that the response time is proportional to the number of LEDs.

Figure 2c shows an improvement over Figure 2b, where the direction of the reflected light beam is detected by a position sensor. The position sensor may be a solid-state image sensor or a special semiconductor chip for detecting the spot position. The response is very quick because the reflected light is detected continuously. However, the position sensor is much bigger than a conventional photo transistor.

In order to detect the orientation of a surface, the range of multiple points must be measured. This can be realized by extending the proximity sensor shown in Figure 2c; multiple LEDs are placed in different positions and excited sequentially. Figure 3 shows an example of the configuration of this type (4). When the spatial coordinates of a spot on the surface for each of six LEDs is obtained, a plane is fitted to the points by a least-squares method.

## BIBLIOGRAPHY

1. S. Tachi, K. Komiya, K. Tanie, T. Ohno, and M. Abe, Guide Dog Robot: Its Basic Plan and Some Experiments with MELDOG MARK III, *Proceedings of the Eleventh International Symposium on Industrial Robots*, Tokyo, 1981, pp. 95–102.
2. A. R. Johnston, "Proximity sensor technology for manipulator end effectors," *Mech. Machine Theor.* **12**, 95–109 (1977).
3. T. Okada, "Development of an optical distance sensor for robots," *Int. J. Robot. Res.* **1**(4), 3–14 (1982).
4. T. Kanade and T. Sommer, An Optical Proximity Sensor for Measuring Surface Orientation for Robot Manipulation, *Robotics Research*, MIT Press, Cambridge, MA, pp. 548–563, 1984.

Y. SHIRAI  
Electrotechnical Laboratory



## QUALITATIVE PHYSICS

Qualitative physics, like conventional physics, provides an account of behavior in the physical world. The vision of qualitative physics, in conjunction with conventional physics, is to provide a much broader formal account of behavior, an account rich enough to enable intelligent systems to reason effectively about it. However, unlike conventional physics, qualitative physics predicts and explains behavior in qualitative terms.

The behavior of a physical system can be described by the exact quantitative values of its variables (forces, velocities, positions, pressures, etc.) at each time instant. Such a description, although complete, fails to provide much insight into how the system functions. The insightful concepts and distinctions are usually qualitative, but they are embedded within the much more complex framework established by continuous real-valued variables and differential equations. Qualitative physics is an alternative physics in which these concepts are defined within a far simpler, but nevertheless formal, symbolic qualitative basis.

It is important to note that qualitative physics yields qualitative descriptions of behavior based on qualitative descriptions of the physical situation and physical laws. The key contribution that makes qualitative physics useful and possible is that moving to the qualitative level preserves the important behavioral distinctions. For example, important concepts and distinctions underlying behavior are state, cause, law, equilibrium, oscillation, momentum, quasistatic approximation, contact force, feedback, etc. These terms are qualitative and can be intuitively understood.

Qualitative physics is, perhaps more than anything, a long-term research program. A great deal of further research is required before qualitative physics can even come close to explaining the range of physical phenomena accounted for by conventional physics. Much of the research on qualitative physics recapitulates fundamental physical and mathematical investigations that took place centuries ago. Driven by the necessity to formalize common sense and enabled by the idea of computation and the modeling techniques of AI, these new theories characterize what the earlier research took for granted.

Qualitative physics is an active area of AI research. Although this research is unified in its goal to account for physical behavior qualitatively, this goal is addressed with a great deal of technical, notational, and methodological diversity. For conciseness, the point of view of Ref. 1 is used, and the alternative proposals are explained in its terms.

**Quantitative vs. Qualitative.** Figure 1 illustrates the approach of qualitative physics contrasted with conventional physics. Both start by modeling physical situations, both end with a qualitative commonsense description of the behavior. The first step in the conventional approach is to formulate and solve the differential equations to obtain a solution. The second step is to interpret this solution to obtain a commonsense description of the behavior. The qualitative analysis begins by formulating qualitative differential equations and then solv-

ing them. The result is a same commonsense description of the behavior that is obtained more simply and a causal explanation for that behavior.

Unlike quantitative variables, qualitative variables can only take on one of a small number of values. Each qualitative value corresponds to some disjoint interval on the real number line. The most important property of a quantity is increase, decrease, or constancy (or, equivalently, its derivative is positive, negative, or zero). This simple, but most important, quantity space consists of only three values: +, -, and 0.

**Why Do Qualitative Reasoning?** The motivations for developing a qualitative physics stem from outstanding problems in psychology, education, AI, and physics. One wants to identify the core knowledge that underlies physical intuition. Humans appear to use a qualitative causal calculus in reasoning about the behavior of their physical environment. Judging from the kinds of explanations humans give, this calculus is quite different from the classical physics taught in classrooms. This raises questions as to what this (naive) physics (qv) is like and how it helps one to reason about the physical world.

In classical physics the crucial distinctions for characterizing physical change are defined within a nonmechanistic framework, and thus they are difficult to ground in the commonsense knowledge derived from interaction with the world. Qualitative physics provides an alternative and simpler way of arriving at the same conceptions and distinctions and thus provides a simpler pedagogical basis for educating students about physical mechanisms.

AI and (especially) its subfield of expert systems (qv) are producing very sophisticated computer programs capable of performing tasks that require extensive human expertise. A commonly recognized failing of such systems is their extremely narrow range of expertise and their inability to recognize when a problem posed to them is outside this range of expertise. In fact, expert systems usually cannot solve simpler versions of the problems they are designed to solve. The missing common sense can be supplied, in part, by qualitative reasoning.

Some of the advantages of qualitative reasoning are:

**It Identifies All Modes of System Functioning.** The qualitative analysis of a system identifies all its possible behaviors. This is crucial in many applications because it highlights undesirable modes to be avoided. This information is almost impossible to obtain with a conventional numerical simulation.

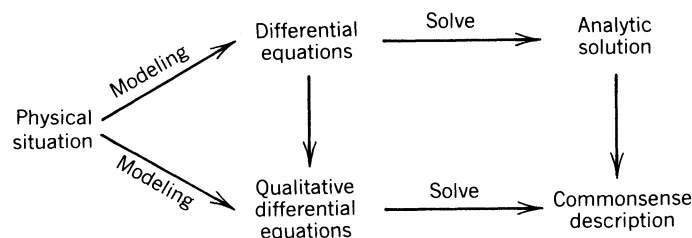


Figure 1. Qualitative vs. quantitative.

**It Functions with Incomplete Models.** A qualitative physics analysis does not require a detailed quantitative model of the system's components—information that may not be easily available or require premature reasoning to determine.

**It Functions with Incomplete Data.** Similarly, a qualitative analysis requires only a qualitative description of the initial conditions and system inputs (if any).

**It Is More Efficient.** Qualitative analysis can be implemented with simple constraint satisfaction (qv), which is much more efficient than symbolic equation manipulation or numerical simulation.

**It Provides Explanations.** Qualitative analysis provides direct, often causal, explanations for its predictions. This information can be conveyed to a user or can be examined by programs to determine what could cause the undesirable behavior (e.g., in design and diagnosis tasks).

**It Simplifies Subsequent Quantitative Analysis.** Qualitative analysis helps identify which component's function is critical and which is secondary. This advice can then be used to guide the selection of quantitative models for components. Also, the solutions to the qualitative equations indicate the regions within which the quantitative solutions are to be found.

**Relationship to Psychology and Physics.** Qualitative physics concerns knowledge about the physical world. It would not be such an important area of investigation if it were not for two crucial facts: (a) people are very good at functioning in the physical world and (b) no theory of this human capability exists that could serve as a basis for imparting it to computers.

The research in qualitative physics shares the goal of providing a framework for understanding the behavior of physical systems. All provide a language to write physical theories (see following sections). However, the goals for these languages differ, resulting in fundamentally different frameworks.

The physical approach (1,2) seeks physical theories that accurately predict behaviors. Although observations of human behavior are used as hints, there is no concern for modeling human foibles. The physical approach seeks to understand the common sense of an ideal expert. As a consequence, this approach adopts some of the methodology of conventional physics and engineering.

The psychological approach (3,4) seeks physical theories that can conform to observed human behavior. This approach seeks to model the behavior of experts as well as neophytes. For example, Forbus (3) provides a physical theory containing the incorrect, but useful, Aristotelian impetus law.

Some research (2,5,7) makes no commitment on this issue.

**Structure and Function.** Qualitative physics defines a relationship between the structure of a physical situation and its function and thus provides the basis for algorithms that determine the function of a device solely from its structure. This aspect is crucial for most applications. For example, if the designer of a troubleshooting program erroneously introduces assumptions of the function of a device into its constituent component models, then the program will miss faults and symptoms or incorrectly identify components as being faulted.

The goal is to draw inferences about the behavior of the composite device solely from laws governing the behaviors of its parts. This view raises a difficult question: Where do the laws and the descriptions of the device being studied come from? Unless one places some conditions on the laws and the descriptions, the inferences that can be made may be (implic-

itly) preencoded in the structural description or the component model library.

The no-function-in-structure principle is central: The laws of the parts of the device may not presume the functioning of the whole. Take as a simple example a light switch. The model of a switch that states "if the switch is off, no current flows; and if the switch is on, current flows" violates the no-function-in-structure principle. Although this model correctly describes the behavior of the switches in our offices, it is false as there are many closed switches through which current does not necessarily flow (such as two switches in series). Current flows in the switch only if it is closed and there is a potential for current flow.

Without this principle, qualitative physics would be an architecture for building hand-crafted (and thus ad hoc) theories for specific devices and situations. It would provide no systematic way of ensuring that a particular class of laws did or did not already have built into them the answers to the questions that the laws were intended to answer.

### Qualitative Physics Analysis

**Structure.** To do a qualitative physics analysis, a program must be provided a description of the physical situation about which it will draw inferences. There are three main approaches: constraint-based, component-based, and process-based.

The constraint-based approach (5,6) describes the physical situation directly in terms of a set of variables and constraints relating those variables. For example, Figure 2 presents the causal-structure description for a simple heat-flow system. The causal structure contains three constraints: a qualitative adder, a direct qualitative proportionality, and a qualitative differential. These constraints relate three variables:  $T$ , the temperature of the material;  $T_s$ , the temperature of the source of heat;  $\Delta T$ , the temperature difference; and inflow, the resulting rate of heat flow into the material.

The process-based approach (3) describes the physical situation in terms of the physical processes potentially present. Intuitively, a process is something that causes changes in objects over time. For example, flowing, bending, heating, cooling, stretching, compressing, and boiling are all processes in qualitative process theory.

The component-based approach, illustrated by Refs. 1, 2,

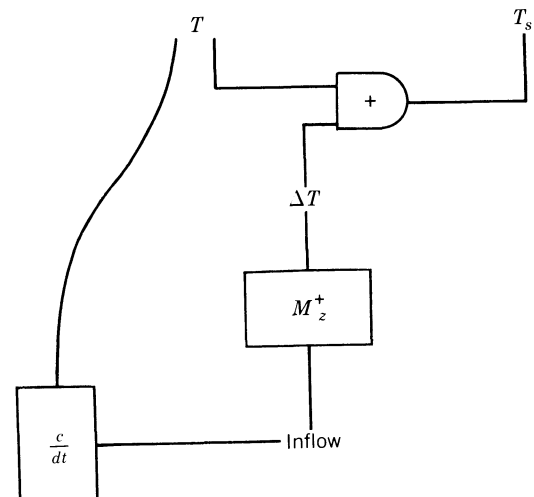


Figure 2. A qualitative causal structure.

and 7, is reductionist: The behavior of a physical structure is accounted for by the behaviors of its physical constituents. Reference 7 describes the situation in terms of the organic molecules present. References 1 and 2 distinguish three kinds of constituents: materials, components, and conduits. Physical behavior is accomplished by operating on and transporting materials such as water, air, electrons, etc. The components are constituents that can change the form and characteristics of material. Conduits are simple constituents that transport material from one component to another and cannot change any aspect of the material within them. Some examples of conduits are pipes, wires, and cables. The physical structure of a device can be represented by a topology in which nodes represent components and edges represent conduits. Figure 3a illustrates the device topology of the pressure regulator diagrammed in Figure 3b. In this device topology the conduits "in" and "out" transport material of type fluid and conduit FP transports material of type force. The control on valve VV adjusts the area available for flow, and SNS senses the output pressure to control the value inversely (so that if pressure rises, area available for flow decreases).

Each modeling approach has its advantages and disadvantages. The constraint-based approach allows one to sidestep the modeling constituents at the cost of ignoring the no-function-in-structure principle. The process-based approach is extremely general, including the capability for creating and destroying constituents as well as rearranging objects. This capability introduces a large amount of inefficiency particularly for systems with large numbers of interconnections (e.g., electrical and fluid circuits). The component-based approach is efficient, more easily obeys the no-function-in-structure principle, and is ideally suited for systems with a fixed interconnect topology.

The naive-physics (qv) (approach 4) does not really fit into the classifications of this entry. It advocates the large-scale

axiomatizations of commonsense domains in predicate calculus. Although the spirit of this proposal guides much of the research, most qualitative physics research deviates from its suggestions. A central focus of qualitative physics is the study of behaviors of systems over time and, in particular, the determination of function from structure—issues about which Ref. 4 is unconcerned.

**Qualitative Arithmetic with +, −, and 0.** The qualitative value of  $x$  is denoted  $[x]$ . Here,  $[x] = +$  iff  $x > 0$ ,  $[x] = 0$  iff  $x = 0$ , and  $[x] = -$  iff  $x < 0$ . Addition and multiplication are defined straightforwardly (Tables 1 and 2):

Table 1.  $[X] + [Y]$

$[X]:$	−	0	+
$[Y]:$			
−	−	−	−
0	−	0	+
+	−	+	+

Table 2.  $[X] \times [Y]$

$[X]:$	−	0	+
$[Y]:$			
−	+	0	−
0	0	0	0
+	−	0	+

Note that although  $[xy] = [x][y]$ ,  $[x + y]$  is not always defined.

This +, 0, − value set is surprisingly versatile. For example, to distinguish  $x$  with respect to landmark value  $a$ , one defines a new variable  $y = x - a$ . Then,  $[y] = +$  corresponds to  $x > a$ ,  $[y] = 0$  corresponds to  $x = a$ , and  $[y] = -$  corresponds to  $x < a$ .

**Qualitative Equations.** A qualitative equation, or confluence, is an expression in terms of qualitative values, variables, and operators. A set of qualitative values satisfies a confluence if either the qualitative equality strictly holds using the arithmetic of Tables 1 and 2 or one side of the confluence cannot be evaluated because the addition operation is not closed. Consider the confluence  $[x] + [y] + [z] = 0$ . If  $[x] = +$  and  $[y] = -$ , this confluence is satisfied for any value of  $[z]$ . A set of values contradicts a confluence if both sides evaluate and the confluence is not satisfied. Thus,  $[x] = [y] = [z] = +$  is contradictory. Note that by this definition a confluence need neither be satisfied nor contradicted if some of the variables do not have assigned values.

**Qualitative Derivatives.** The statement, " $x$  is increasing" in the formalism is  $[dx/dt] = +$ . This notation tends to be cumbersome both for typography and computer I/O. Thus,  $\partial x$  is used as an abbreviation for  $[dx/dt]$ , and more generally  $\partial^n x$  for  $[d^n x/dt^n]$ . Note that unlike in quantitative calculus,  $\partial^{n+1} x$  cannot be obtained by differentiating  $\partial^n x$  (e.g., consider  $x = 2 + \sin t$ ,  $[x] = +$  everywhere but  $\partial x$  varies between +, 0, and −). One always has to go back to the quantitative definition:

$$\partial^{n+1} x = [d^{n+1} x / dt^{n+1}]$$

At present, there is little notational agreement in qualitative physics research. Although Refs. 1, 9, and 2 use the conventions in this entry, in Ref. 10  $[x] = + \equiv X = +$ , in Ref. 3  $[x] = A_s x$  and  $\partial x \equiv D_s x$ , in Ref. 5  $\partial x = + \equiv$  increasing ( $x$ ), and in Ref. 11  $[x] = +$  and  $\partial x = +$  is represented as  $\langle (0, \infty), \text{inc} \rangle$ .

**Quantity Spaces.** The +, −, 0 value space is insufficient for many applications. Typically, a variable has distinguished values above or below which behavior is radically different. For example, a string breaks if its tension is too high ( $T > T_{\max}$ ). Thus, a network of inequalities may exist among the variables. As the number of such inequalities is finite, the set

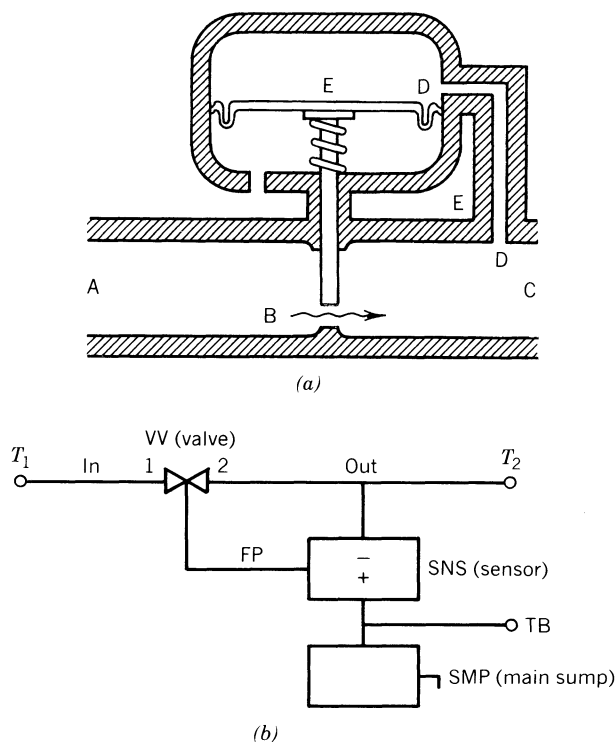


Figure 3. Pressure regulator.

of all inequalities divides the potential values for each variable into a finite set of intervals on the real line. Each such region corresponds to a particular qualitative value. Thus, in the case of a string,  $T$  has (at least) two values: below  $T_{\max}$  and above  $T_{\max}$ .

The choice of distinguished values for variables is a serious problem. Clearly, 0 is an important landmark for derivatives, but what is the origin of the distinguished values for other variables? One possible scheme is to choose simple symbolic vocabularies (e.g., tall, very tall, etc.). Such arbitrarily chosen schemes are based on the particular situation being analyzed. It is always possible to choose just the "right" symbolic vocabulary for each variable after analyzing the situation. This produces a model, a solution, and an interpretation that have the appearance of cogency but are, in fact, vacuous as the appearance of success depends on the context-sensitive choice of distinguished values.

Most implementations require extensive computation with inequalities. Reference 3 utilizes a general quantity-space representation to reason with inequalities in qualitative-process theory.

**Ambiguities.** Qualitative reasoning is inherently ambiguous. Qualitative equations cannot be manipulated as conventional quantitative ones. The conventional manipulations rely on the mathematical field axioms; however, Table 1 does not describe an algebraic group. For example, one cannot subtract  $[y] + [z] = 0$  from  $[x] + [y] + [z] = 0$  to obtain  $[x] = 0$  because the assignment  $[x] = +$ ,  $[y] = +$ , and  $[z] = -$  also satisfies the two original confluences.

Due to this ambiguity, qualitative reasoning may make multiple behavioral predictions (called interpretations). At a minimum for a prediction to be correct, one of the interpretations must correspond to the actual behavior of the real system. A stronger criterion follows from observing that the structural description of a particular device implicitly characterizes a wide class of physically realizable devices with the same topology. The stronger criterion requires that the behavior of each device in the class is described by one of the interpretations, and every interpretation describes the behavior of some device in the class.

Unfortunately, this criterion is too strong (12). There are devices that have interpretations that are not physically realizable. (Fortunately, in practice, the criterion is almost always obeyed.) For example, if the quantitative model was  $x + y + z = 0$  and  $y + z = 0$ , the confluences would be  $[x] + [y] + [z] = 0$  and  $[y] + [z] = 0$ . The qualitative operations of Tables 1 and 2 place no constraints on the value of  $[z]$ . The original quantitative model indicates that  $x = 0$ , but this cannot be inferred from the confluences alone. The same two confluences also describe the quantitative equations  $x + 2y + z = 0$  and  $x + y = 0$ , from which it is not possible to infer  $z = 0$ .

**Modeling.** In the process- or component-based approach qualitative analysis must construct a qualitative model consisting of constraints from the structural description. (This step is avoided in constraint-based approaches in which the physical situation is initially described in terms of constraints.)

In the component-based approach each type of physical constituent has a distinct model. The component model characterizes all the potential behaviors that the component can manifest. The lawful behavior of a component is expressed by a set

of possible states and their associated specifications and confluences. For example, a valve could be modeled:

Open:  $[A = A_{\max}]$ ,  $[P] = 0$ ,  $\partial P = 0$

Working:  $[0 < A < A_{\max}]$ ,  $[P] = [Q]$ ,  $\partial P + [P]\partial A - \partial Q = 0$

Closed:  $[A = 0]$ ,  $[Q] = 0$ ,  $\partial Q = 0$

where  $A$  is the area available for flow,  $P$  is the pressure across the valve, and  $Q$  is the flow through the valve. In the open state the valve functions as a simple conduit, there is no pressure drop across it, and the flow through it is unconstrained. Neither can the pressure across it change—that can only be caused by a change in position of the valve. The closed state is the dual to the open state. In it the valve completely disconnects the input from the output. There is no flow through the valve, and the pressure across it is unconstrained. The flow through it cannot change without changing the area available for flow. In the working state the valve acts like a fluid resistance—its resistance controlled by  $A$ . For example, if  $\partial A = 0$ , then  $\partial P = \partial Q$ .

A state specification consists of a set of inequalities that defines different component-operating regions whose behavior is governed by distinct confluence sets. These inequalities play a fundamental role in reasoning with time because variables will change enough to cause components to change state.

The component models for a given composite device state (i.e., a selection of operating regions for all the device's components) define a set of confluences that govern the behavior of the device in each composite state. Any complete set of variable values that satisfies all confluences for all the components of the system specifies a possible behavior.

The laws for flowlike variables are based on the assumption that conduits are always completely full of material and incompressible, and therefore any material added through one terminal must result in material leaving by some other terminal. In other words, material is instantaneously conserved. This rule is called the continuity condition from system dynamics (13). For electrical circuits it is called Kirchhoff's current law, for fluid systems the law of conservation of matter, and for thermal systems the law of conservation of energy. The continuity condition requires that the sum of the current, forces, heat flows, etc., into a conduit (and most components) be zero. As these rules are simple sums, they apply to the derivatives of attributes also. These can all be expressed as confluences.

As the value of the pressurelike variable in a conduit is the same everywhere in the conduit, there are no pressure laws for individual conduits or components. No matter which path the material takes from component A to B, the sum of the individual pressure drops along each path must be equal. This is called the compatibility condition from system dynamics. For example, if the pressure between conduits A and B is  $x$  and the pressure between conduits B and C is  $y$ , the pressure between conduits A and C is  $x + y$  (and thus, qualitatively,  $[x] + [y]$ ). Compatibility requires equal voltages, velocities, etc., at points where the components are connected.

The process-based approach takes a mirror-image approach to modeling. Instead of modeling the lawful behavior of objects, a process describes possible interactions among a collection of objects. For example, Figure 4 is the specification of a heat-flow process. In qualitative process theory, a process is specified by five parts: the individuals, preconditions, quantity conditions, relations, and influences. Heat flow applies to

## Process heat-flow

## Individuals:

src an object, Has-Quantity(src, heat)  
 dst an object, Has-Quantity(dst, heat)  
 path a Heat-Path, Heat-Connection(path, src, dst)

## Preconditions:

Heat-Aligned(path)

## Quantity conditions:

$A[\text{temperature}(\text{src})] > A[\text{temperature}(\text{dst})]$

## Relations:

Let flow-rate be a quantity  
 $A[\text{flow-rate}] > \text{ZERO}$   
 $\text{flow-rate} \propto (\text{temperature}(\text{src}) - \text{temperature}(\text{dst}))$

## Influences:

$1 - (\text{heat}(\text{src}), A[\text{flow-rate}])$   
 $1 + (\text{heat}(\text{dst}), A[\text{flow-rate}])$

Figure 4. Heat-flow process.

three individuals: src, the source of the heat; dst, the destination of the heat; and path through which the heat flows. The preconditions must hold for the process to be active. The preconditions are intended to reflect external conditions (affected by an external agent). A precondition for heat flow is that the heat path must be aligned so that heat can flow. The quantity conditions must hold for the process to be active. For example, heat can only flow if the temperature of the source is higher than the temperature of the destination. Quantity conditions differ from preconditions in that quantity conditions only test internally affected quantities. The relations imposes constraints among the parameters of the individuals. For example, the amount of the flow rate is greater than zero and proportional to the difference between the source and destination temperature (an indirect influence). The direct influences specify the root causes that change parameters. Here  $I^-(\text{heat}(\text{src}), A[\text{flow rate}])$  specifies that the flow rate negatively influences the amount of heat in the source. The development and clarification of this language is a major contribution of Ref. 3.

At any particular time a quantity must be directly influenced, indirectly influenced, or not influenced at all. If a quantity is both directly and indirectly influenced, the physical theory is incorrect. This condition is intended to capture a notion of causality. All directly influenced quantities are independent, acted on directly by active processes. All other quantities must be changed indirectly as a consequence of changes processes make on the directly influenced quantities.

The direct influences on a quantity are combined to determine its derivative:

$$\partial x = \sum_{\text{processes}} I(x, q)$$

using the qualitative arithmetic of Tables 1 and 2. If the sum is ambiguous, inequality information is used to resolve influence. For example,  $\partial x$  if the only direct influences on  $x$  are  $I^-(x, a)$  and  $I^+(x, b)$  (where  $a > 0$  and  $b > 0$ ). However, if  $a > b$ , then  $\partial x = -$ . The indirect influences are resolved by propagating out the values of the directly influenced quantities. This propagation is simple because the network of qualitative proportionalities is assumed to be loop-free.

The process-based and component-based approaches are

similar in that they both rely on an underlying formulation in terms of constraints and conditions. However, information that is locally available in one theory is distributed in the other. In this sense they are duals to each other. In qualitative-process theory the influences of diverse processes must be gathered to determine a constraint on a variable. In the component-based approach each confluence of a component model places a necessary constraint on all the variables it references.

Processes can become active and inactive as quantities change. This feature is used to great advantage in developing physical theories. For example, the three behavioral states of the pressure regulator would be modeled by three different processes. However, sometimes differing processes are required in the process-based approach when one component suffices in the component-based approach. For example, an additional instance of the heat-flow process is required if heat could flow in either direction while the same component set models heat flow in both directions.

**Qualitative Calculus.** As time passes, variables change toward thresholds causing transitions to other operating regions and component states (equivalently, processes become active and inactive). At any given time, many variables may be approaching their respective thresholds, and the analysis must determine which variable(s) reach their thresholds first. This process is sometimes called limit analysis (3) or transition analysis (2).

A theory of change presumes a theory of time. Roughly speaking, three models of time have been used in qualitative physics. References 3 and 7 model time as a sequence of intervals as suggested by Ref. 14. Reference 1 models time by intervals separated by any number of instants. References 2, 5, and 9 model time as intervals separated by single instants. This latter approach is used in the remainder of this entry.

The rules necessary to reason about change over time derive directly from the conventional calculus (2,15), particularly the mean-value theorem. These rules, in essence, solve the qualitative differential equations constructed by the modeling. If variables are differentiable, the rules for determining time behavior are remarkably simple. The following five rules apply to all derivative orders:

- A Instant-to-interval.** Any nonzero quantity must remain nonzero during the following interval. If a quantity  $[x]$  is zero at the instant, on the following interval it must obey the integration constraint  $[x] = \partial x$ .
- B Interval-to-instant.** A nonzero quantity may become zero on the following interval iff  $[x] = -\partial x \neq 0$ .
- C Continuity.** Variables change continuously. Continuous changes are between 0 and + or - (in either direction) but not between + and -.
- D Contradiction avoidance.** A transition is only possible if the resulting state satisfies the qualitative equations for that state.
- E Analyticity.** A quantity that is zero for any interval is zero for all time. Conversely, a quantity that is nonzero at some time cannot become identically zero.

**Envisioning.** Envisioning is the reasoning process based on the five rules of the qualitative differential calculus to determine the temporal behavior of the system:

1. Start with some initial state(s) (usually an instant).
2. Identify those quantities that are moving to their thresholds.
3. Construct partial descriptions of tentative next states using rules A or B.
4. Using rules C, D, and E, expand and prune the possible next states.
5. For each state not yet analyzed, go to step 2.

Note that envisioning can proceed in parallel. The resultant state diagram is the envisionment for the system. In order to obtain a comprehensive understanding of system function, the envisionment includes all possible initial states.

The envisioning process can be illustrated by examining the diaphragm-spring-stem fragment of the pressure regulator. If the input pressure increases, the output pressure increases, producing a force on the diaphragm. This force acts against the spring force and friction. The valve slowly gains velocity as it closes; however, by the time it reaches the position where the force exerted by the pressure balances the restoring force of the spring, the valve has built up a momentum causing it to move past its equilibrium position, thus reducing the pressure below what it should be. As it has overshoot its equilibrium, the spring pushes it back; but by the same reasoning the valve overshoots again, thereby producing ringing or oscillation. Figure 5 illustrates the essential details; a mass situated on a spring and shock absorber (i.e., friction).

The behavior of the mass is described by Newton's law,  $F_m = ma$ , or, qualitatively,  $[F_m] = \partial v$  (more generally,  $\partial^n F_m = \partial^{n+1} v$ ). Hooke's law for the spring,  $F_s = -kx$ , becomes  $\partial^{n+1} F_s = -\partial^n v$ . The resistance of the shock absorber is modeled by  $\partial^n F_f = -\partial^n v$ . For simplicity sake, define  $x = 0$  as the mass position with the spring at equilibrium and  $x > 0$  to be to the right. The net force on the mass is provided by the spring and shock absorber:  $F_m = F_s + F_f$ , or, qualitatively,  $\partial^n F_m = \partial^n F_s + \partial^n F_f$ .

Suppose the system is started by stretching the mass to the right. At this instant the velocity is zero, but the mass is pulled to the right. Thus, the initial instant is

$$[x] = +, \partial x = 0, \partial^2 x = -, [F_m] = -, \partial F_m = +, \\ [F_f] = 0, \partial F_f = +, [F_s] = -, \partial F_s = 0$$

By rule A all nonzero quantities remain nonzero on the following interval:

$$[x] = +, \partial x = ?, \partial^2 x = -, [F_m] = -, \partial F_m = +, \\ [F_f] = ?, \partial F_f = +, [F_s] = -, \partial F_s = ?$$

In addition, by rule A the instantaneously zero quantities, particularly  $[F_f]$  and  $\partial x$ , must obey the integration constraints on the interval, and as their respective derivatives are known:

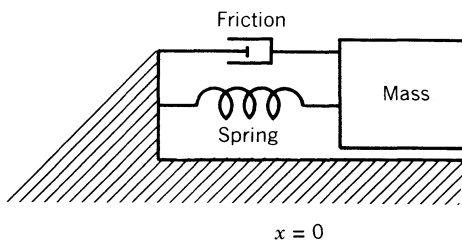


Figure 5. Mass-spring-friction system.

$[F_f] = \partial F_f = +$  and  $\partial x = \partial^2 x = -$ . If no further derivatives are computed, rule A places no constraint on  $\partial F_s$ . Superficially it looks as if the initial state can change to one of three possible subsequent states, corresponding to  $\partial F_s = +, -, \text{ or } 0$ . However, using rule D, the only consistent value for  $\partial F_s$  is  $+$ . Therefore, the transition is unambiguous:

$$[x] = +, \partial x = -, \partial^2 x = -, [F_m] = -, \partial F_m = +, \\ [F_f] = +, \partial F_f = +, [F_s] = -, \partial F_s = +$$

The analysis for the transition to the subsequent instant is more complex but nevertheless unambiguous as well. By rule B the only variables guaranteed not to change are  $\partial x = -$  and  $[F_f] = +$ . However, those quantities determine the values of the remaining variables. By Hooke's law,  $\partial F_s = +$ . Using rule D, consider the possibility that  $[F_s]$  becomes zero. Then,  $[F_m] = [F_f] + [F_s] = +$ . However,  $[F_m]$  cannot change from  $-$  to  $+$  without violating continuity rule C. Thus,  $[F_s] = -$ . Following this style of argument, the next state is unambiguous:

$$[x] = +, \partial x = -, \partial^2 x = 0, [F_m] = 0, \partial F_m = +, \\ [F_f] = +, \partial F_f = 0, [F_s] = -, \partial F_s = +$$

Applying rules A, C, and D produces a single unambiguous subsequent interval:

$$[x] = +, \partial x = -, \partial^2 x = +, [F_m] = +, \partial F_m = +, \\ [F_f] = +, \partial F_f = -, [F_s] = -, \partial F_s = +$$

The only possibility here is  $\partial F_m$  from  $+$  to  $0$  to  $-$ .

This generation process continues 18 steps until the initial situation is reencountered. The resulting envisionment shows that changes in forces cause changes in acceleration, which cause changes in velocity, which cause changes in position, which in turn causes changes in forces producing an endless oscillation. The qualitative physics analysis has provided an accurate description of the behavior of the mass-spring system without ever invoking the classical differential calculus.

## Current and Future Research

Qualitative physics is a research direction that has only just started. Although significant progress has been made, relatively few physical phenomena have been accounted for. The following are a few of the current directions of qualitative physics research.

Most of the qualitative physics research presumes that the physical structure can be modeled by a relatively small number of distinct objects and interactions. However, many physical situations are, modeled by a large number of identical objects: water flow in a river, heat flow along a slab, molecules in a pipe, etc. Such systems cannot be accounted for in current qualitative physics. Qualitative physics has, thus far, only examined lumped-parameter systems described by ordinary differential equations (but see Ref. 7). Little progress has been made on distributed systems whose behavior is described by partial differential equations with respect to spatial variables.

Qualitative physics has tended to focus on dynamics. Reasoning about spatial movement of objects and shapes of objects and their interactions is extremely difficult and only preliminary progress has been made (16,17). Reference 18 illustrates some of the difficulties.

A much more sophisticated notion of time is required. Thus far, time has been an implicit, not explicit, variable, making it



difficult to express certain laws (delays) and to reason about the consequences of the events without analyzing every intermediate event as well (but see Ref. 8).

Systems oscillate, approach asymptotes, and become unstable. Most qualitative physics cannot account for such long-term behavioral effects.

Most qualitative physics has assumed that the underlying functions are well behaved, continuous, and differentiable. Often systems go through discontinuous transitions that are produced by or produce impulses. A qualitative theory of such generalized functions is required.

Humans must learn commonsense physics from interactions with the world. Reference 19 provides some suggestions about learning qualitative laws.

Complex systems can be described at many levels of abstraction. Currently, qualitative physics does not include any notion of hierarchy, or when it is necessary to move to other levels of abstraction.

### Literature Survey

The collection (Ref. 20) includes many of the papers referenced in the bibliography and is the best general introduction to the subject. A broad range of papers on common sense and naive physics is Ref. 21. Two early papers on qualitative reasoning and processes are Refs. 22 and 23. Two early publications on qualitative physics are Refs. 16 and 24. Reference 2 has extensive discussions of the relation to conventional calculus and provides a very different approach to transition ordering than the one presented here. In Ref. 5 new qualitatively distinguishable points (landmarks) are created dynamically. References 1 and 8 present much of the material in this entry in far greater detail. Reference 1 introduces a notion of mythical causality that explains qualitative predictions. Reference 18 and 25 explore the role of ambiguity in detail. Reference 26 provides a truth-maintenance facility specifically adapted for the kinds of ambiguities that arise in qualitative reasoning.

Qualitative physics techniques have been applied to a variety of domains: electric circuits (23,25); VLSI (2); medicine (27); turbojet engines (6); and molecular genetics (7). Reference 28 proposes to apply qualitative physics to the operation of companies. Reference 29 proposes to use qualitative reasoning in circuit design. Reference 30 critiques Ref. 1 and contrasts the techniques of qualitative physics with those of comparative statics. Refs. 30 and 31 argue that there is no causality in feedback loops. Reference 32 is a reply to Ref. 30, which discusses causal ordering in detail and contrasts the divergent views of economics and engineering on reasoning about systems.

### BIBLIOGRAPHY

1. J. de Kleer and J. S. Brown, "A qualitative physics based on confluences," *Artif. Intell.* **24**(1-3), 7-83 (1984).
2. B. C. Williams, "Qualitative analysis of MOS circuits," *Artif. Intell.* **24**(1-3), 281-346 (1984).
3. K. D. Forbus, "Qualitative process theory," *Artif. Intell.* **24**(1-3), 85-168 (1984).
4. P. J. Hayes, The Naive Physics Manifesto, in D. Michie (ed.), *Expert Systems in the Microelectronic Age*, Edinburgh University Press, pp. 242-270, 1979.
5. B. Kuipers, "Commonsense reasoning about causality: Deriving behavior from structure," *Artif. Intell.* **24**(1-3), 169-203 (1984).
6. R. Rajagopalan, Qualitative Modeling in the Turbojet Domain, in *Proceedings of the National Conference on Artificial Intelligence*, Austin, TX, pp. 283-287, 1984.
7. D. S. Weld, The use of aggregation in causal simulation, *Artificial Intelligence* **30**(1), 1-34 (1986).
8. B. C. Williams, Doing time: Putting qualitative reasoning on firmer ground, *Proc. of the Fourth AAAI Conf.*, Philadelphia, PA, 1986, pp. 105-112.
9. J. de Kleer and D. G. Bobrow, Higher-Order Qualitative Derivatives, in *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, 1984, pp. 86-91.
10. J. de Kleer and J. S. Brown, The Origin, Form and Logic of Qualitative Physical Laws, in *Proceedings Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, 1983, pp. 1158-1169.
11. B. Kuipers, The Limits of Qualitative Analysis, in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 128-136, 1985.
12. B. Kuipers, "Qualitative simulation," *Artif. Intell.* **28**(3), 289-338, (1986).
13. J. L. Shearer, A. T. Murphy, and H. H. Richardson, *Introduction to System Dynamics*, Addison-Wesley, Reading, MA, 1971.
14. J. Allen, Maintaining Knowledge about Temporal Intervals, TR-86, Computer Science Department, University of Rochester, Rochester, NY, 1981.
15. B. C. Williams, The Use of Continuity in a Qualitative Physics, in *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, pp. 350-354, 1984.
16. J. de Kleer, Qualitative and Quantitative Knowledge in Classical Mechanics, Artificial Intelligence Laboratory, TR-352, MIT, Cambridge, MA, 1975.
17. K. D. Forbus, Qualitative Reasoning about Space and Motion, in D. Gentner and A. Stevens (eds.), *Mental Models*, Erlbaum, Hillsdale, NJ, 1983.
18. D. S. Weld, Issues in the Automatic Description of Complex Mechanical Devices, Working Paper, Xerox, PARC, 1985.
19. K. D. Forbus and D. Gentner, Learning Physical Domains: Towards a Theoretical Framework, in *Proceedings of the Second International Machine Learning Workshop*, pp. 53-73, 1983.
20. D. G. Bobrow, *Qualitative Reasoning about Physical Systems*, MIT Press, Cambridge, MA, 1985.
21. J. R. Hobbs and R. C. Moore, *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, 1985.
22. G. Hendrix, "Modeling simultaneous actions and continuous processes," *Artif. Intell.* **4**, 145-180 (1973).
23. J. S. Brown, R. R. Burton, and F. Zdybel, "A model-driven question-answering system for mixed-initiative computer-assisted instruction," *IEEE Trans. Sys. Man Cybernet.* **3**(3), 248-257 (1973).
24. J. de Kleer, Causal and Teleological Reasoning in Circuit Recognition, Artificial Intelligence Laboratory, TR-529, MIT, Cambridge, MA, 1979.
25. J. de Kleer and J. S. Brown, Assumptions and Ambiguities in Mechanistic Mental Models, in D. Gentner and A. S. Stevens (eds.), *Mental Models*, Erlbaum, Hillsdale, NJ, 1983, pp. 151-190.
26. J. de Kleer, "An assumption-based truth maintenance system," *Artif. Intell.* **28**(2), 36 (1986).
27. B. J. Kuipers and J. P. Kassirer, "Causal reasoning in medicine: Analysis of a protocol," *Cog. Sci.* **8**, 363-385 (1984).
28. P. E. Hart, A. Barzilay, and R. O. Duda, Qualitative Reasoning for Financial Assessments: A Prospectus, *AI Mag.* (1986).
29. B. C. Williams, Principled Design Based on Qualitative Behavioral Descriptions, MIT Artificial Intelligence Laboratory, 1985.

30. Y. Iwasaki and H. A. Simon, "Theories of causal ordering—reply to de Kleer and Brown," *Artif. Intell.* **29**(1), 63–72 (1986).
31. H. A. Simon, Causal Ordering and Identifiability, in W. C. Hood and T. C. Koopmans (eds.), *Studies in Econometric Models*, New York, Wiley, 1953; also in *Models of Discovery*, by H. A. Simon, D. Reidel, Boston, 1977.
32. J. de Kleer and J. S. Brown, "Theories of causal ordering," *Artif. Intell.* **28**(3), 33–62 (1986).

J. DE KLEER  
Xerox PARC

## QUESTION ANSWERING

Question answering (Q/A) in AI has, historically, been a branch of natural-language processing (NLP). It has served two complementary roles: a context in which to look at basic issues in natural-language understanding (qv) and natural-language generation (qv) and an application area, in the form of natural-language interfaces, to database systems and more recently help systems and expert systems (qv). Of course, many of the desirable features of natural-language Q/A can and should be carried over to more formal user–system interactions.

### Theoretical Framework

A question is a request by the speaker for the respondent either to inform the speaker of some information or perform some act. This difference can be illustrated by the two questions "Can you speak Spanish?" and "Can you please speak Spanish?" The former is, ambiguously, either a request for information about the speaker's abilities or a request for a performance, whereas the latter can only be a request for a performance. An answer is the information or performance directly requested. (This has been called elsewhere a "direct answer.") A response comprises the respondent's complete informative and performative reaction to the question, which can include elements of each of the following: an answer, additional information or actions that are salient to the answer, information or actions instead of an answer, and additional information provided or actions performed that are salient to this substitute for an answer. Making this distinction between answers and responses and among the various potential components of a response allows a clean separation between the information or action the question attempts to elicit, the goal that the questioner attempts to satisfy through asking the question, and the material that the respondent feels is necessary or useful to contribute to the interaction.

To see that responses can include elements of each of the four types noted above, consider the following question:

Q. What's the combination of the printer-room lock?

Response R1 below is simply a direct answer.

R1. It's 8-9-8.

Response R2 contains both an answer and an additional salient action that will show the questioner how to apply the answer.

R2. It's 8-9-8 (Simultaneously keying in the numbers to show how)

R3 contains both an answer and additional salient information, i.e., that the answer should not be taken as holding correct indefinitely (which might otherwise have been assumed).

R3. It's 8-9-8, but Ira changes it every week.

R4 below does not contain an answer but rather two kinds of information instead of an answer, i.e., first, rejection of the question ("I don't know therefore I'm not the right person to have asked") and then a suggestion as to alternative source of the requested information.

R4. I don't know. Why don't you ask Ira?

R5 just contains an action instead of an answer, which presumably cannot be revealed to the questioner.

R5. (Respondent keys in the numbers without revealing them to the questioner)

And finally, R6 just contains information instead an answer.

R6. The door's open.

Notice that Q/A per se does not necessarily confer the role of questioner (Q) on a user and the role of respondent (R) on the system. Expert systems (qv) are known to need and request additional information from their users during the course of an interaction, although so far a user's ability to reply to such questions has been limited to giving a number or choosing a menu item.

### Questions

If the answer part of a response, if present, constitutes the information or performance directly requested, it is useful to consider what information or performance questions generally request. (The examples come primarily from transcripts of naturally occurring interactions, with others taken from existing Q/A systems. In each case, Q labels the participant—user or system—who is asking the relevant question and R the participant who is responding.)

First, one can question whether a statement is true. Such questions are most often used to obtain information, to elicit a performance, or to verify that one has understood something correctly:

Q. Now on that 1040, do I just file my income from this job?  
R. No. You have to put all of your income.

Q. I see—now the money-market fund and the all-saver's certificate, they're not the same are they?

R. No—oh no—a money-market fund is a mutual fund, an all-saver's certificate is a certificate that you get from your bank which is tax free for federal purposes up to two thousand of interest.

In both responses, "no" constitutes the answer, with additional goal-related information provided in the first response and justification provided in the second.

Second, one can question which individuals, sets, actions, events, etc., satisfy some (possibly partial) description:

Q. Could I just have that toll free number?

R. It's 1-800-424-1040.

Q. Who has to file an estimated tax return?

R. Anyone who has an income that they're not paying withholding tax on.

Both responses contain simply an answer.

Third, one can question the meaning of some term. Such questions are often used to learn about new concepts, to understand what the respondent means by a term or phrase that the questioner understands in a particular way, or to obtain clarification of some utterance:

R. In your agreement of sale, make sure that you have an engineering-inspection clause.

Q. Engineering-inspection clause?

R. Right. You want that house inspected by an engineer just to make sure that there's nothing wrong with it that you might have missed. The clause stipulates that an engineer be permitted at your expense to inspect the house and that if there is anything in that house that requires a repair (and you can set the amount, \$250 or \$500), the owner will be responsible for it, or else you will be permitted to get out of the deal.

This response contains both a definition of the term "engineering-inspection clause" and justification of R's advice to have such a clause.

Fourth, one can ask "Why is it the case that P?" expecting some explanation (qv) of P in terms of the events leading up to it (example 1); the "standing conditions" that enable events to lead to P (example 2); or the actor's motivation for doing P (example 3):

Example 1:

Q. Why is this conductor warped?

R. By mistake, J closed the switch, shorting the circuit. So some of the components got damaged.

Example 2:

Q. Why is this conductor warped?

R. The earth's magnetic field is not negligible, exerting a strong force at right angles to the conductor in the horizontal plane. This puts a strong acceleration on the conductor, making it warp.

Example 3:

Q. Why? (i.e., Why did you pick up the green pyramid?)

R. To get rid of it.

Q. Why did you do that?

R. to clean off the red cube.

Q. Why did you clean off that cube?

R. to put it on a large green cube?

Q. Why did you do that?

R. to stack up the red cube, a large red block, and a large green cube. (1)

Explanation in expert systems (2,3) and database systems enriched with reasoning (qv) capabilities (4,5) has focused, so far, on systems explaining aspects of their behavior: why they requested some information from the user and why they drew a particular conclusion. Given their reasoning abilities, the only facts systems have been able to identify and present as relevant to an explanation are reports of actions they have taken or are taking and, only more recently, the underlying standing conditions justifying these actions (3) and the strategic decisions organizing them (6). They may yet improve in their explanatory abilities with respect to:

The kinds of things that users will be able to request explanations of, e.g., "What difference is my answer going to make?" "Why should I follow your advice?" "What if I don't follow it?" etc.

The kinds of relevant facts (and theories relating facts) that the system can draw upon in forming explanations; e.g., it should be able to explain its advice either in terms of how it came to that advice or in terms of the expected benefits to the user in following it.

Sensitivity to the contextually appropriate alternatives to what it has been asked to explain—allowing Q to explicitly indicate alternatives, as above, or alternatively, using its beliefs about Q's expectations and interests to identify appropriate alternatives.

Finally, in this partial list of question types, one can ask whether two descriptions or statements are (roughly) paraphrases. Such questions are often used to verify that one has correctly understood something:

R. Okay, under those circumstances I would not object to see you go back into that for another six months.

Q. So you roll it over, in other words.

R. Right.

Q. That's what I'm looking at, Schedule A.

R. Okay. If you turn it over, there is the Schedule B.

Q. Oh, it's on the back of Schedule A?

R. Right.

## Answers

There are two important things to understand about answers. First, while there may be many correct answers to a question, only a few of them may be useful, and it can make a significant difference to the questioner which of them is given. Second, even a correct and useful answer may be misunderstood or misleading, thereby causing the person receiving the answer as much of a problem as an incorrect or useless one. To be cooperative, an answer must be correct, useful, and nonmisleading. This is best shown in answers to questions about the referent of some description.

**Correct Answers.** The correct answer to a question about the referent of some description is usually taken to be an enumeration or description of all and only those individual entities that satisfy the questioned description. Computing a correct answer is not a problem in standard database systems, all of which model a domain with a finite number of known individuals and relations. In other cases it is more difficult. First,

if a knowledge base reflects a full first-order theory, there is no way to guarantee that an answer is complete, i.e., that the system can identify all the relevant individuals. For any individual it may be impossible to prove that it does or does not satisfy the questioned description. Moreover, if a knowledge base admits descriptions of some entities in terms of other ones (usually through functions), it admits the possibility of an infinite number of entities (implicitly) in the knowledge base satisfying the questioned description. In addition, the same entity might be included more than once in an answer under alternate descriptions (7,8).

Second, if a knowledge base is purely intensional (i.e., it models a collection of concepts whose extension is immaterial), there is no fixed set of terms denoting individuals (e.g., The Admiral Nimitz, the JFK, etc.), with other terms denoting sets of those individuals (e.g., ships, aircraft carriers, etc.). The decision as to whether a concept should be treated as an individual or as a subsuming abstraction (9) depends on the given question. For example, the question

Q. List the foreign cars

could be answered taking manufacturer as individuator,

R1. BMW, Toyota, Saab, Fiat . . .

or taking manufacturer and model as individuator,

R2. BMW318, BMW325, BMW533, Toyota Tercel, Toyota Celica, Toyota Camry, Saab 99, Saab 900, Fiat X1/9 . . .

or taking manufacturer, model, and year as individuator,

R3. '83 BMW318, '85 BMW325, '84 BMW533, '80 Toyota Tercel, '84 Toyota Celica, '84 Toyota Camry, '78 Saab 99, '85 Saab 900, '76 Fiat X1/9. . . .

Here there are several correct answers. However, for the answer to the follow-up question

Q. Which of them have over a 2-liter engine?

to be correct, it must be given at the level of R2 or R3, since engine size depends on model. That is, the R1-level individual "BMW" from the previous answer can neither be included in nor excluded from the current answer because the 318 has a 1.8-liter engine, while the 325's engine is 2.5 liter.

Third, if a knowledge base represents only as much as it truthfully knows (i.e., it contains disjunctive, existential, and negative assertions as well as specific and universal assertions), it may only be able to give indefinite answers (7) to *wh*-questions. For example:

Q. What city is the capital of California?

R1. San Francisco or Sacramento.

R2. An inland city in California.

R3. A California city other than LA.

Levesque (8) proposes an enriched interface language to respond more appropriately in such cases of more general variation in a knowledge base system's knowledge of the world.

Last, if a knowledge base is dynamic (i.e., it is aware of how it can change over time), the problem with correct answers

comes from modeling a dynamically changing process: the system may know what the correct answer was for time point  $T_i$  and may know that it expects to learn of a new correct value corresponding to time point  $T_{i+1}$ . When the question is asked (i.e., NOW), it may be between points. Thus, the system does not have an answer that it knows to be correct for NOW. All it can do is respond with information that is salient to the answer (10). For example:

Q. Where's the Kitty Hawk?

R. At 0430 it was at (longitude/latitude). I will have an update of that position at 1630.

**Useful Answers.** Consider the following interchange, adapted from Ref. 11:

Q. Where is Joyce Chen's?

R1. It's in Cambridge, MA.

R2. It's at 330 Ringe Avenue.

R3. It's a half block from the Fresh Pond station on the Red Line on the other side of Fresh Pond Parkway.

R4. Over there.

All the answers correctly specify the location of Joyce Chen's but are not equally useful. Answer R1 would be useless if the user were already in Cambridge trying to find the restaurant, whereas answer R2 would be equally useless if the questioner did not know the streets of Cambridge and was wondering whether to get there by a cab, subway, or walk. Answer R3 would be useless if the questioner were within sight of Joyce Chen's, whereas R4 would be useless if the questioner were trying to ascertain whether drinks were available (given Cambridge is wet, while its immediate neighbors, Belmont and Arlington, are dry).

Early AI work on useful answers was done by Lehnert (12) in terms of assigning questions to conceptual categories which, in turn, constrain the range of acceptable answers. More recently, useful answers have been discussed in terms of Q's plans and how Q sees the requested information as enabling those plans. The most relevant work in this area is Cohen's work (13) on planning (qv) speech acts (qv) needed to achieve some goal; Appelt's work (14) on planing utterances to achieve goals, where that planning uses a formal logic of knowledge and action to take account of the effect of actions on knowledge; and Allen's work (15) on inferring plans and goals from appropriate queries and Pollack's work (16) on inferring plans and goals from possibly inappropriate ones.

**Nonmisleading Answers.** It is not enough for an answer to be correct and useful: It can still cause problems by misleading the questioner if he or she expects the answer to be more informative than it is and hence interprets it as such:

Q. Which ships have a doctor on board?

R. The JFK, the Admiral Nimitz . . . .

Q concludes that these ships have a doctor on board and whatever others there are do not. That is, Q is led to believe that there is a nonempty set of ships that do not have a doctor on board. However, if there are only these ships and no others, the respondent R's answer, although correct and useful, is nevertheless misleading: R should have answered "all of them,"

which would not have misled Q about the nonemptiness of this other subset of ships.

The principles that guide this behavior are Grice's maxim of quantity:

*Make your contribution as informative as is required for the current purposes of the exchange. Do not make your contribution more informative than is required for the current purposes of the exchange* (17).

and Joshi's modification to Grice's maxim of quality:

*If you, the speaker, plan to say anything which may imply for the hearer something that you believe to be false, then provide further information to block it* (18).

What Joshi's modified quality maxim provides is a criterion for the level of informativeness needed to satisfy the quantity maxim: Enough must be said so that the hearer does not draw a false conclusion.

Although the above example does not require the respondent to augment his response beyond a simple answer, most often adhering to these principles does require this, e.g.,

Q. How many ships have a doctor on board?

R. All 43 of them.

where, besides the numerical answer requested by "how many," R indicates that they exhaust the set. This behavior is therefore discussed further in the next section.

## Responses

Answers, of themselves, are not enough for effective communication, which seems to rely on R responding to a question with more than (or other than) an answer. One can categorize the component information included in responses in terms of what function each component is meant to serve. The following classification is not exhaustive. Moreover, the same component may be able to fulfill more than one function. Nevertheless, it is useful to consider each category separately.

1. The answer.
2. Additional information (or an offer to provide information) that R believes that Q may need in order to fulfill Q's plan.
3. Information (or an offer to provide information) that R believes that Q may need to satisfy Q's goal (at some level), where the plan that R believes Q to have will not fulfill that goal.
4. Additional information that R believes justifies or explains that response, so that Q will accept it.
5. Additional information or graphical presentation that R believes will clarify that response, so that Q will understand it.
6. Information intended to correct a misconception that Q reveals through Q's question, that interferes with R's ability to answer that question or with Q's ability to correctly interpret R's answer.
7. Information intended to correct a misconception that Q reveals through his question, that does not interfere with R's ability to answer that question or with Q's ability to

correctly interpret that answer, but that R feels responsible for correcting.

8. Information that R believes that Q may need to keep from misconstruing R's answer. (That is, Q's question reveals no misconception, but R suspects that Q may have partial or incorrect information that may lead Q to draw a false conclusion from R's answer.)
9. Additional information (or an offer to provide such information) that R believes that Q needs for Q's general knowledge.
10. Information as to R's attitude toward the answer.
11. Rejection of the question (often accompanied by support for such a seemingly uncooperative action).

With respect to this classification, what have been called "indirect answers" (19) do not constitute a separate category. In an indirect answer the answer is meant to be inferred from the information given explicitly. However, this information will be one (or more) of the types mentioned above; for example:

Q. Would you like some more chicken?

R. I'll have some more carrots. (information relevant to Q's perceived goal of serving seconds)

Q. Would you like some more chicken?

R. I'm full, thank you. (information that explains R's answer, so that Q will accept it)

For this reason, indirect answers do not constitute a separate type of response but may be found as examples illustrating the various functions that are served by the component information in a response.

## Goal-Related Information and Offers: Appropriate Plans.

Many types of response behavior can be attributed to R's attempt to infer and address Q's plans and not just Q's explicit question. In cases where R believes Q's plans are appropriate to Q's goals, he or she may respond not only to the obstacle represented by the given question but also to any remaining obstacles detected in the plan to enable Q to overcome them (15):

Example 1:

Q. When does the Windsor train leave?

R. 4 P.M., Gate 7.

Example 2:

Q. Does the Ottawa train leave at 4 P.M.?

R. No, it leaves at 5 P.M.

Allen's system detects obstacles in the course of inferring the questioner's plan and goal. For example, in example 1 the explicitly indicated obstacle to Q's presumed plan of boarding the Windsor train is Q's lack of knowledge of its departure time. Moreover, Q's potential lack of knowledge of the train's departure location is an additional obstacle that could prevent Q's boarding that train: hence, the gate information is provided as well. In example 2 Q's goal of boarding the Ottawa train is blocked by Q's explicitly indicated lack of knowledge of its departure time. Allen rightly comments that if the answer

to Q's question were "yes," this obstacle would be eliminated, but a "no" answer leaves Q with the obstacle intact: The additional information provided about the departure time is meant to eliminate it.

Similar behavior has also been recognized and implemented with moderate success in simpler systems that do not have elaborate plan-recognition machinery. In Kaplan's COOP system (20) a natural-language database interface, if the answer to a user's question came back "no" or empty, COOP checked to see if the question reflected particular misconceptions about the database (see below). If there was no misconception—the answer as just negative or empty—COOP would attempt to suggest relevant information that might be requested in a follow-up query by broadening the "focus" of the original question and answering the new question. This he called a "suggestive indirect response." In example 3 (below) the additional information in R1 is an answer to the question "John is an X: class-level" and R2 an answer to the question "X: person is a senior." In example 4 the focus of the question is changed from "specialties in High Altitude Observatory" to "specialties," and the answer to the derived question is included in the response:

Example 3:

Q. Is John a senior?  
R1. No, he's a junior.  
R2. No, but Fred is.

Example 4:

Q. What specialties in High Altitude Observatory do programmers from SYS have?  
R. I don't know, but you might be interested in any specialties that programmers from SYS have: (list of specialties)

Kaplan acknowledges that determining the focus of a question cannot really be done purely syntactically but requires recognition of the questioner's plans. However, the behavior that COOP displays is useful and requires very little additional computational machinery.

HAM-ANS (21) is another system that attempts to provide Q with additional information beyond that requested: It will either provide additional case role fillers, as in example 5, or use more specific quantifiers, as in example 6:

Example 5:

Q. Has a yellow car gone by?  
R. Yes, one yellow one on Hartungstreet.

Example 6:

Q. Have any vehicles stopped on Biberstreet?  
R. Yes, two.

Both types of information are ones that HAM-ANS determines anyway in the course of answering the user's question and so require no substantial additional computational burden. Since CO-OP provides this kind of extended response in the case of a negative answer and HAM-ANS does so for positive answers, the two techniques for anticipating follow-on questions complement each other.

Monitor offers are another way of addressing Q's plans and goals:

Example 7:

Q. Has flight AA57 landed yet?  
R. No, it's still circling. Do you want me to let you know when it lands?

The question in R's response constitutes an offer to monitor for and provide Q with additional information, when and if R learns of it. Mays (10) has identified the knowledge and reasoning capabilities needed by a system in order to offer competent monitors, i.e., monitors for an event that might actually occur. (It would be uncooperative to offer monitors for events that could not occur.) The complementary problem of deciding which of the many possible competent monitors to offer is addressed by Cheikes (22), based on Allen's work—reasoning about the questioner's plans and potential obstacles to those plans. In cases where monitor offers will turn out to be appropriate, however, there are two kinds of obstacles, ones that R can do something about, through information he or she provides (example 1) or actions taken (e.g., opening the door for someone burdened with packages) and obstacles that R has no control over. The former can be dealt with as Allen does, but the latter R must be able to reason about, to discover whether they may somehow, sometime become unblocked. If they will become unblocked, R must be able to reason about whether Q could carry out the plan at that point, or whether other constraints on Q's plan make it impossible. (R might then suggest or help Q come up with a different plan to achieve Q's goal or an acceptable alternative, as discussed in the next section.)

The final example is an indirect answer:

Example 8:

Q. What time do you want to leave?  
R. I must be in San Diego before 10 A.M.

This example comes from a dialogue between GUS (23), a very early toy expert system, and a user attempting to arrange a flight to San Diego. Q has already found out from R what day R wants to go. What example 8 communicates indirectly is a rejection of the question "I don't know what time I want to leave." Thus, knowing R's desired departure time remains an obstacle to Q's goal of arranging a flight for R. What is communicated directly, however, is information that R believes is relevant to Q's overcoming this obstacle: Since desired departure time is related to desired arrival time, constraints on the latter will propagate back to constraints on the former. Since users do not always know more than just some constraints on an answer, expert systems should be able to deal with these responses.

#### Goal-Related Information and Offers: Inappropriate Plans.

Unlike the examples in the previous section, it is often the case that Q has a desired goal but that the plan he comes up with, which motivates the question, will not achieve that goal. If R recognizes this, he may respond with information that facilitates Q's achieving the desired goal, as well as (directly or indirectly) answering Q's given plan-related question. This kind of behavior is illustrated in the following examples taken from Ref. 16:

Example 9:

Q. Is there a way to tell mail that I'd like to deliver a message after some particular time and/or date?



R. No. I can't really see any easy way to do this either (and guarantee it). If you really wanted to do that, you could submit a batch job to run after a certain time and have it send the message.

Example 10:

Q. Is there any way to tell a bboard (see Blackboard systems) to load only new messages? (I would like to speed up entry into infomac).

R. Mail cannot do this because it parses the entire message file whenever you start up . . . I will clean out the informac bboard so that it will load faster.

Pollack (16) addresses the question of inferring inappropriate plans underlying queries, based on a model of responding to questions that does not rest on what she has called the appropriate query assumption. Abandoning this assumption requires the development of a model of plan inference that is significantly different from the models discussed by Allen (15), Cohen (13), and others. Pollack's procedure involves R's reasoning about likely differences between R's own beliefs and Q's beliefs and searching through the space of plans that could be produced by such differences. Her notion of plans derives from the literature in the philosophy of action, especially Refs. 24 and 25.

**Information Intended To Justify or Explain R's Response.** For brevity, this discussion is limited to questions as to whether a statement is true. The kind of information given in response to justify or explain yes-no answers ranges from a description or enumeration of the simple facts supporting a yes answer to an existential question (i.e., "Do any X's Y?") to the general principles, concept definitions, and specific facts supporting any answer to any yes-no question. Knowing the grounds for an answer as simple as yes or no can make that response much more acceptable.

Facts provided as grounds justifying a yes answer to an existential yes-no question were illustrated in the HAM-ANS examples (5 and 6). When asked about the existence of objects in an image (or events in a sequence of images) that satisfy a particular description, it looks for and records all such instances. Thus, it can easily enumerate or describe the instances to justify its positive response. (This is one case where the same information can fulfill more than one function.)

For yes-no questions in general, where the answer may depend on the meaning of the concepts involved, a response may include a justification in the form of a definition of the relevant concepts or a comparison of them, as in the following example.

Example 11:

Q. I see; the money-market fund and the all-saver's certificate, they're not the same are they?

R. No, oh no, a money-market fund is a mutual fund, an all-saver's certificate is a certificate that you get from your bank that is tax free for federal purposes up to two thousand of interest.

Work done by McKeown (26) on automatically generated answers to requests for the definition of a database term or for a comparison of two terms, based on an enriched database model, is directly applicable to providing this type of justification.

Where the system's answer to a yes-no question involves reasoning from general principles as well as from specific facts, that reasoning can be described as a way of justifying R's response:

Example 12:

Q. Is Bill citizen-of x.

R. Yes.

Q. Why?

R. I can show

Bill citizen-of UK

by using

x citizen-of y if x born-in y

and I know

Bill born-in UK

The example comes from an interface to a PROLOG-based reasoner called "Query-the-User" (27), which keeps a record of its proof tree. Because PROLOG (see Logic programming) is a Horn-clause-reasoning system (i.e., all its rules are of the form  $A_1 \& A_2 \& \dots \& A_n \Rightarrow B$ , where all literals are positive), its proofs constitute reasonably comprehensible justifications. However, Horn-clause logics have severe limitations since they cannot express disjunctive facts and conclusions and true negation. But sound proof methods for stronger logics may bear little or no resemblance to any type of human reasoning, and texts produced directly from their proofs may make little or no sense. The point is not that one cannot produce justifications of yes-no answers if a system uses an effective but unnatural proof procedure but rather that additional work must be done on ways of mapping from general proof structures to comprehensible ones, whose translation into natural language makes sense.

"Should" yes-no questions come up in the context of advisory systems (i.e., as opposed to diagnostic systems). There are many grounds for advising courses of action—goals to be advanced for the various parties involved, potential risks to the various parties, costs to be paid—and factors must both be identified and weighed. Justification is as important to include in response here as an answer. But notice that there are also two other possible answers when decisions depend on weighing factors: "yes and no" if the evidence seems to balance and "it doesn't matter" if both courses of action lead to the same result:

Example 13:

Q. Should I take AI this semester?

R1. No. Although you've had the prerequisites for AI, you have many required courses left to complete.

R2. Yes. If you take it now, you'll be able to take more of the advanced topics in AI later.

R3. Yes and no. Although you've had the prerequisites for AI, you have many required courses left to complete. On the other hand, if you take it now, you'll be able to take more of the advanced topics in AI later. (28)

Example 14:

Q. Should I turn off the faucet?

R. It doesn't matter. I've already shut off the main. (19)

McKeown (28) has done preliminary work on justifying advice in the domain of student advising, where the advice can de-

pend on both domain requirements (e.g., Physics 150 is required for a BSE degree) and user preferences (e.g., The user/student likes linguistics courses). She shows how the same action can be advised in favor of and advised against, depending on which requirements and preferences are taken into consideration.

**Information Intended to Clarify R's Response.** Two obvious types of information that can be used to clarify what one's answer is are examples and pictures. Examples can be used to pin unfamiliar concepts and abstract descriptions to things one's audience is familiar with, and, as has been noted (29), they allow one to explore the limits of generalizations and explanations. Pictures, on the other hand, address a different mental processor than text and thus enable an information provider to illustrate the same information in another way.

Computational work on constructing examples and integrating them into responses to requests for on-line help has been done by Rissland and her students at the University of Massachusetts (29). Their on-line help system links a (primarily canned) textual explanation program with an example generator that generates examples on the fly using a knowledge base of existing examples (organized into a taxonomy) together with procedures of modifying and instantiating them according to the user's needs and/or desires and according to the current context. Particular calls to HELP result in the selection of a scriptlike structure (a TEXPLATE) of pointers to chunks of text and/or requests for explicit examples or constraints describing modifications to be made to an example. A TEXPLATE will also set up a context for interpreting future requests to HELP, e.g., a request for help with "queue" will be interpreted in the context of a previous request for help with "print":

Example 15:

Q. help 'print'

R. PRINT queues one or more files for printing, either on a default system printer or on a specified device.

For example, PRINT LOGIN.COM will cause your file login.com to be printed on the default printer.

Q. help 'queue'

R. A print queue is a list of files to be printed on a specific device.

For example, PRINT LOGIN.COM / QUEUE=NARROW requests that file login.com be printed on the printer corresponding to the queue named 'narrow.'

The /QUEUE qualifier is synonymous with the /DEVICE qualifier. For instance, PRINT LOGIN.COM / DEVICE=NARROW will do the same thing.

An interpreter controls the flow through the TEXPLATES, including which user options to present and what to do in response to a user's request for further explanation or additional examples.

With respect to pictures used to clarify text, there has been little research so far on systems that can reason about how information should be presented, text or figures alone or some combination of the two. [Although STEAMER (30) uses annotated diagrams to illustrate cause-and-effect relationships between steam-room (power-plant) controls and the parts of the hydraulic systems they affect, the text and pictures are

canned: Steamer does not reason about what to present.] These are not picture database systems, which accept requests for the set of pictures satisfying a given description or systems which allow a user to ask questions about a given picture (21,31). Rather these are systems that will reason about optimal presentation of information in terms of pictures or text or both.

**Information Intended To Correct a Misconception.** Many of the troubles in interaction come from the fact that participants hold different views about the world and either do not recognize it or fail to do anything about it. As a result, each may leave the interaction with very different beliefs about what was communicated.

In simple cases users may be able to recognize a disparity between the system's view of the world and their own, e.g., if the system's answer seems wrong or strange. On the other hand, users of knowledge-based systems will have less knowledge of the system's area of expertise. Thus, users are less likely to be able to recognize a disparity. By not recognizing the disparity they may misinterpret or at best be simply confused by the system's response. Therefore, systems must be able to recognize and respond to the misconceptions the user reveals.

Computational work on recognizing and responding to "existential" misconceptions, such as in example 16, has been done by Kaplan (20) and Mercer (32). Computational work on recognizing and responding to intensional misconceptions (i.e., misconceptions about what can serve as an argument to a particular relation), such as in example 17, have been done by Mays (33):

Example 16:

Q. Which French majors failed CIS531 last term?

R1. I do not know of any French majors.

R2. I do not know of CIS531 being given last term.

Example 17:

Q. Which grad students have taught CIS531?

R. CIS531 is a grad course. Only faculty can teach grad courses.

Other computational work has been done on responding to object-related misconceptions (34). These may be revealed through a user's utterance either describing an object in terms of a class it does not belong to, incorrectly attributing some property to it, or ascribing an incorrect value to one of its properties:

Example 18:

Q. What's the interest rate on this stock?

R. Stocks don't have an interest rate. They may pay a dividend periodically.

In this work McCoy has concentrated on the role of discourse perspective: first, in deciding whether something reflects a misconception in the current context (which highlights certain attributes of an object and suppresses others) and, second, in deciding what information to include in an appropriate response (i.e., what would be an appropriate correction in the current context).

There are clearly many other types of misconceptions that

can interfere with R's ability to answer Q's questions or with Q's ability to interpret the answers correctly. Yet misconception corrections, where called for, are an important component of responses, so more work should be done in this area.

**Information Intended To Prevent Misconstrual.** The problem of avoiding potentially misleading utterances was introduced above. Such behavior was guided by Joshi's modification to Grice's maxim of quality (see above under Nonmisleading Answers):

*If you, the speaker, plan to say anything which may imply for the hearer something that you believe to be false, then provide further information to block it* (18).

To conform to this requires that R be able to anticipate when Q might draw a false conclusion from R's true answer and that R be able to figure out what to say to block it. R's anticipatory reasoning should nonetheless not lead to a computational explosion. Research in this area has so far identified the following bases for such reasoning: the implicatures of answering in a particular way (35), users' default beliefs about the world (36), and expectations that a cooperative respondent will address one's underlying goals and not just one's surface request (37).

**Rejection of the Question.** Given the problems with answering and responding to questions appropriately, not much attention has been paid to enabling systems to justifiably refuse to answer a user's question. This is a problem that will have to be faced as database and knowledge-based systems become more selective as to whom they provide particular information to (i.e., issues of privacy and security) and such systems become aware of both their own and other systems' capabilities, thereby becoming able to offer competent "referrals." The interface aspects of this behavior are not the hard part: The hard part involves computing whether users should be provided with particular information or whether it will enable them to infer things that they are not privileged to and giving a system awareness of its own capabilities and those of other systems so that it can determine whether it can provide the requested information or, if not, what system can. Notice that the latter is related to the same kind of reasoning involved in describing the user and the system's mutual beliefs about what services the system can provide, which is needed in order to understand and respond to the user's plan-related questions.

## Conclusion

Q/A is not necessarily an interaction that must be carried out in natural language, but the features of natural-language Q/A presented above must be available for any successful interaction of person and machine.

## BIBLIOGRAPHY

1. T. Winograd, *Understanding Natural Language*, Addison-Wesley, Reading, MA, 1972.
2. R. Davis, "Interactive transfer of expertise," *Artif. Intell.* **12**, 121-157 (1979).
3. W. Swartout, "XPLAIN: A system for creating and explaining expert consulting programs," *Artif. Intell.* **21**, 285-325 (1983).
4. C. Kellogg and L. Travis, Reasoning with Data in a Deductively Augmented Data Management System, in H. Gallaire (ed.), *Advances in Data Bases*, Plenum, New York, 261-295, 1981.
5. A. Walker, Automatic Generation of Explanations of Results from Knowledge Bases, Technical Report RJ3481, IBM San Jose Research Laboratory, May 1982.
6. D. Hasling, W. Clancey, and G. Rennels, "Strategic explanations for a diagnostic consultation system," *Intl J. Man-Machine Stud.* **20**, 3-20 (January 1984).
7. R. Reiter, Deductive Question-Answering in Relational Data Bases, in H. Gallaire and J. Minker (ed.), *Logic and Data Bases*, Plenum, New York, pp. 149-178, 1978.
8. H. Levesque, "A functional approach to representation," *Artif. Intell.* **23**, (2), 155-212 (1984).
9. F. Corella, Semantic Retrieval and Levels of Abstraction, in L. Kerschberg (ed.), *Proceedings of the First International Workshop on Expert Database Systems*, Kiawah Island, SC, October 1984, pp. 397-420.
10. E. Mays, A Temporal Logic for Reasoning about Changing Data Bases in the Context of Natural Language Question-Answering, in L. Kerschberg (ed.), *Expert Database Systems: Proceedings of the First Intl. Workshop on Expert Database Systems*, Kiawah Island SC, October 1984, pp. 238-257.
11. G. Grewendorf, What Answers can be Given? in F. Kiefer (ed.), *Questions and Answers*, D. Reidel pp. 45-84, 1983.
12. W. Lehnert, A Computational Theory of Human Question Answering, A. Joshi, B. Webber, and I. Sag (eds.), in *Elements of Discourse Understanding*, Cambridge University Press, Cambridge, UK, pp. 145-176, 1981.
13. P. Cohen, On Knowing What to Say: Planning Speech Acts, Technical Report 118, Department of Computer Science, University of Toronto, January 1978.
14. D. Appelt, *Planning English Sentences*, Cambridge University Press, Cambridge, UK, 1985.
15. J. Allen, Recognizing Intentions from Natural Language Utterances, M. Brady (ed.), in *Computational Models of Discourse*, MIT Press, Cambridge, MA, pp. 107-166, 1982.
16. M. Pollack, Information Sought and Information Provided, *Proceedings of CHI'85* (Computer-Human Interfaces), Association for Computing Machinery (ACM), San Francisco, CA, pp. 155-160, April 1985.
17. H. P. Grice, Logic and Conversation in P. Cole and J. L. Morgan (eds.), *Syntax and Semantics*, Vol. 3, Academic Press, New York, pp. 41-58, 1975.
18. A. K. Joshi, Mutual Beliefs in Question Answering Systems, in N. Smith (ed.), *Mutual Belief*, Academic Press, New York, pp. 181-221, 1982.
19. J. Hobbs and J. Robinson, "Why ask?" *Disc. Proc.* **2**, 311-318 (1979).
20. J. Kaplan, Cooperative Responses from a Portable Natural Language Database Query System, in M. Brady (ed.), *Computational Models of Discourse*, MIT Press, Cambridge MA, pp. 167-208, 1982.
21. W. Wahlster, H. Marburger, A. Jameson, and S. Busemann, Over-Answering Yes-No Questions: Extended Responses in a NL Interface to a Vision System, *Proceedings of the Eighth IJCAI*, Karlsruhe, FRG, August 1983, pp. 643-646.
22. B. Cheikes, Monitor Offers on a Dynamic Database: The Search for Relevance, Technical Report CIS-85-43, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, October 1985.
23. D. Bobrow, R. Kaplan, D. Norman, H. Thompson, and T. Winograd, "GUS, A frame driven dialog system," *Artif. Intell.* **8**, 155-173 (1977).
24. A. Goldman, *A Theory of Human Action*, Prentice-Hall, Englewood Cliffs, NJ, 1970.

25. M. Bratman, Taking Plans Seriously, *Social Theory and Practice*, Vol. 9, pp. 271–287, 1983.
26. K. McKeown, *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*, Cambridge University Press, Cambridge, UK, 1985.
27. M. Sergot, A Query-the-User Facility for Logic Programming, in P. Degano and E. Sandewall (ed.), *Integrated Interactive Computing Systems*, North-Holland, Amsterdam, pp. 27–41, 1983.
28. K. McKeown, M. Wish, and K. Matthews, Tailoring Explanations for the User, *Proceedings of Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 794–798, August 1985.
29. E. Rissland, E. Valcarce, and K. Ashley, Explaining and Arguing with Examples, *Proceedings of the Fourth National Conference on Artificial Intelligence*, University of Texas at Austin, pp. 288–294, August 1984.
30. J. Hollan, E. Hutchines, and L. Weitzman, “STEAMER: An interactive inspectable simulation-based training system,” *AI Mag.* 5, (2), 15–28 (Summer 1984).
31. J. Conklin and D. McDonald, Salience: The Key to the Selection Problem in Natural Language Generation, *Proceedings of the Twentieth Annual Meeting of the Association for Computational Linguistics*, University of Toronto, pp. 129–135, June 1982.
32. R. Mercer, and R. Rosenberg, Generating Corrective Answers by Computing Presuppositions of Answers, not of Questions, *Proceedings of the 1984 Conference*, Canadian Society for Computational Studies of Intelligence, University of Western Ontario, London, Ontario, pp. 16–19, May 1984.
33. E. Mays, Failures in Natural Language Systems: Application to Data Base Query Systems, *Proceedings of the First National Conference on Artificial Intelligence*, Stanford, CA, pp. 327–330, August 1980.
34. K. McCoy, Correcting Object-Related Misconceptions, Technical Report MS-CIS-??, University of Pennsylvania, Department of Computer and Information Science, Philadelphia, PA, 1985, Ph.D. Thesis.
35. J. Hirschberg, Scalar Implicature, Ph.D. Dissertation, University of Pennsylvania, Philadelphia, PA, December 1985.
36. A. Joshi, B. Webber, and R. Weischedel, Living Up to Expectations: Computing Expert Responses, *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin TX, pp. 169–175, August 1984.
37. A. Joshi, B. Webber, and R. Weischedel, Preventing False Inferences, *Proceedings of COLING-84* (International Conference on Computational Linguistics), Stanford, CA, pp. 134–138, July 1984.

B. WEBBER  
University of Pennsylvania

# R

## REASONING

Reasoning in general refers to several different kinds of activity. Among the more common of these in humans are drawing conclusions from a set of facts, diagnosing possible causes for some condition, making assumptions about a situation, analyzing and organizing the facts and data concerning some problem, solving a problem or puzzle, and arguing with or persuading another person to a particular point of view. Computer programs exist for all but the last of these. Reasoning forms the basis for all human mental activity and is similarly the basis, in one form or another, for most AI programs. In contrast to simple inference (qv), reasoning usually involves rather long sequences of individual, small inferences organized so as to address a main goal or problem, although the two terms “inference” and “reasoning” are often used interchangeably.

Computer programs that embody some form of reasoning evolved almost as early as modern digital computers themselves. Among the earliest AI programs were the game-playing (qv) programs of the 1950s (1). The use of a game-tree (qv) look-ahead with a static evaluation function to rank the resulting potential game situations can be thought of as a simple kind of reasoning about different alternatives for the progress of the game. These techniques became highly refined during the 1960s. Another important kind of reasoning on the computer that began in the late 1950s was logical reasoning based on mathematical logic (qv) (2–4). This area of research, called

theorem proving (qv), became a major topic in the 1960s (5–7) and had a major impact on the development of many related areas such as expert systems (qv), etc. Another kind of reasoning, means-ends analysis, developed during the 1960s with the study of such systems as GPS (8). An important change of direction occurred in the late 1960s, namely considerable emphasis was given to special-purpose reasoning systems in which special heuristics (qv) and techniques over and above the standard, general ones could be applied to a specific problem type. Examples are the systems that proved properties of programs that developed from the late 1960s (9) and the various expert systems that were developed early (10) and continue to be actively studied and developed to date. Several other important types of computer reasoning developed during the 1970s, imprecise reasoning (11), inductive reasoning (12), and later, default reasoning (13,14). Many of these special kinds of reasoning are described in detail in other entries in this encyclopedia. (e.g., see Circumscription; Expert systems; Meta-knowledge, meta-rules, and meta-reasoning; Reasoning, causal; Reasoning, default; Theorem proving). This entry describes general features exhibited by most reasoning systems.

### Classification of Reasoning Methodologies

There are several different, useful schemes for classifying the types of reasoning activity. Among the more important ones are the degree of precision, the level of reasoning, and the degree of generality.

**Precise versus Imprecise Reasoning.** One classification scheme involves the "precision" with which reasoning steps are made. At one end of this spectrum are reasoning processes that are logically valid. A simple example is *modus ponens*, i.e., from  $p \rightarrow q$  and  $p$  conclude  $q$ . The result is always true in any interpretation in which the premises are true. Various computer-based rules of inference satisfy this property, among them resolution, paramodulation, factoring, and the like (see Resolution; Theorem proving). Reasoning systems built around such rules of inference are typically used for problems that are of a more precise nature such as proving theorems in mathematics, proving properties of computer programs, deriving programs, designing digital circuits, etc. The same kinds of logical inference rules are used as the basis for logic-programming systems, like PROLOG (see Logic programming), and these systems have been used for a variety of problems including intelligent databases and natural-language parsing (qv) and have also been used as general programming languages.

At the other end of the precision spectrum are "fuzzy" reasoning systems (see Fuzzy logic). Such systems attempt to deal with inexact concepts like "a fairly tall person." Even if the height of each person in, say, inches is known, the idea that someone is "a tall person" is not precisely definable. In fact, it may even depend on the context of the reasoning. A particular pigmy may be relatively tall in his own tribe but relatively short in comparison with, say, Americans. Another variation of precision occurs in systems in which each rule or base statement has an attached probability or plausibility. The inference rules themselves then have two parts, one that forms the conclusion from the hypotheses and one that calculates a probability factor for the conclusion based on the probabilities of the hypotheses (15). Thus, the conclusions drawn may not be absolutely true in the system being modeled but may occur with a probability factor hopefully close to the probability attached to those conclusions. Many expert systems, especially those used for diagnosis, use this kind of reasoning (see Ref. 15 and also Expert systems). For example, in medical diagnosis (see Medical advice systems) a particular set of symptoms need not logically imply a disease (i.e., the disease need not always be present when the symptoms are), but the disease may occur a given percentage of the time, the percentage being determined by analyzing many previous cases with those symptoms. Of course, such systems may also include rules specifying definite causal relationships by assigning the probability factor of 1. Such rules would then be like the logical rules in that the conclusion from a set of such rules should always be true when the hypotheses are.

**Problem Level versus Meta-level Reasoning.** A second classification is based on the level at which the program is reasoning, reasoning within the problem domain itself or reasoning about the problem. For example, a program might try to determine the sequence of actions in order for a robot to build a tower from a set of blocks by simulating sequences of such actions in its internal representation of the robot's environment. The program might try all sequences of two actions, then all sequences of three actions, etc., or more likely it would use some heuristics for exploring the space of such sequences. If the program operates by repeatedly selecting some sequence according to some kind of selection criterion and extends the chosen sequence, the program is performing problem-level search. On the other hand, the program might perform an

analysis like the following: Any action that moves the robot out of the circle surrounding the blocks to be used cannot help in building the tower; therefore, never extend an action sequence in such a way that the robot moves outside this area. The first part of this reasoning process does not directly explore the space of actions but does serve to redirect that exploration by imposing new criteria for selecting which sequences to extend or how to extend them. This is reasoning about the problem, i.e., metareasoning. It is substantially more difficult to set up a program to do metareasoning as well as problem reasoning because the user or system designer must provide facts and rules at the metalevel about the facts and rules and situations of the problem itself. These metafacts and metarules are usually not as easy to quantify and/or symbolize (see Ref. 16 or Meta-knowledge, meta-rules, and meta-reasoning).

A more recently explored kind of metareasoning involves making assumptions in the absence of contrary information. A typical example is the class of "negation-as-failure" rules in logic programming (see Logic programming). There, the failure to prove  $p$  is used as justification for assuming  $\neg p$ . This is metareasoning because the formula derived, in this case  $\neg p$ , is not directly derivable in first-order logic, the basis for logic programming. As above, a higherlevel process draws a conclusion from some property of a lowerlevel process, in this case the absence of a proof within the given resources. The main purpose of such metareasoning is to avoid the need to include negative information and the "only-if" halves of definitions explicitly. Consider, e.g., an information system with facts about parents and rules defining grandparent, ancestor, etc. If such a system were required to store all instances of people, say  $A$  and  $B$ , such that  $A$  was not  $B$ 's parent, the size of the system would increase tremendously. Similarly, showing that  $A$  was not a grandparent of  $B$  in the absence of this kind of metarule would involve an arduous deduction using the "only-if" half of the definition of grandparent. With the metarule, the system can simply note that  $P(A, B)$  does not occur and conclude  $\neg P(A, B)$ . This kind of metarule is used extensively in deductive databases and is called the closed-world assumption (CWA) (17). Techniques of the type described in Circumscription and Reasoning, default in this work attempt to delineate the usage of rules that make such assumptions.

**General-Purpose versus Special-Purpose Systems.** A third classification of reasoning systems is based on the degree of generality of the system: Can the system be reasonably applied to a wide class of problems or is it fine-tuned to be effective on only one type of problem? On one end of this spectrum are the general-purpose theorem provers (see Theorem proving; Resolution). These systems typically use some form of logic to represent problems and use very general inference rules. They have been applied to problems ranging from mathematics to information retrieval in databases to circuit design and program generation (6,17,18). At the other end of the spectrum of generality are various expert systems whose representation schemes, inference rules, and search strategies have been optimized for one particular problem domain (10,19). In between these two extremes are reasoning tools that allow some generality while providing the ability to tailor the inference system to the user's problem. Examples are LMA for theorem proving (20), OPS5 (qv) and other expert-system tools (21), and the various production systems (22).

### Reasoning Systems

A reasoning system has several basic components, among them a representation scheme (e.g., logic, semantic nets, frames, etc.), a set of inference rules (e.g., resolution, frame instantiation, etc.), and some means for controlling the way the system applies the inference rules in searching for solutions. Examples of reasoning systems are theorem provers (20), using logic as the representation with resolution, paramodulation, etc., as the inference rules and various strategies like set-of-support and weighting to guide the search, production systems (22), using rules as the representation with chaining as the inference rule and heuristics like chaining from the most instantiated literal, and default systems (14), where the representation might be similar to logic, and one of the inference rules is to conclude the relevant default value for an attribute in the absence of evidence to the contrary (see also Frame theory; Logic; Reasoning, default; Resolution; Rule-based systems; Semantic networks; Theorem proving).

**Controlling the Application of Inferences.** An important aspect of reasoning systems is the kind of control or guidance that is available to direct the search for an answer. Such control mechanisms are referred to as search strategies and are crucial in most applications because the number of conclusions that can be drawn in any given situation is vastly larger than the number that will participate in a solution to the problem at hand. For example, a robot trying to plan how to build the tower could draw thousands of conclusions about moving around the room that would not be relevant to building the tower. Several control strategies are more widely used. Among the most widely used are forward and backward reasoning (also often called forward and backward chaining). In the representation of a problem, some of the formulas represent given facts about the situation (e.g., where the blocks are located that are to be used to build the tower), others represent general rules and facts about the universe of the problem (e.g., how does the act of lifting change the state of the universe, how much weight can the robot lift), and finally some formulas represent the goal configuration. The reader should be aware that the term "rule" is used in many different contexts. An inference rule is a method for deriving conclusions from existing statements, as, e.g., *modus ponens* above. In production systems, expert systems, and others, a rule is a formula in implicative form, i.e.,  $h_1, \dots, h_n \rightarrow c$ . The hypotheses  $h_i$  and the conclusion  $c$  are usually quite general, and such a rule is often written in an "if-then" form. For example, a robot rule might be, "IF (robot is at position  $x$ ) & (box is at position  $x+1$ ) & (box is on the floor) THEN (box is at position  $x$  and off the floor after the robot picks it up)." Here, box and  $x$  can refer to arbitrary boxes and positions. Facts can be thought of as rules with no hypotheses. For example, "box 1 is at position 3." As mentioned above, some facts are general and describe the universe of interest. For example, the amount of weight the robot can lift is probably independent of any task the robot is given to do. On the other hand, there are facts that may be special depending on the task. Thus, the position of the blocks is especially relevant to the task of building a tower but may be of little importance to some other task.

In backward reasoning, the program starts with the goal configuration, finds one or more general rules whose conclusion matches the goal to some extent, and formulates as a new goal the problem(s) of satisfying all the conditions for applying

the general fact. For example, the goal in the tower problem for the robot might be to have three blocks piled one on top of the other. Part of this configuration is to have block  $b$  on top of block  $a$ . Suppose there were a rule of the form "IF (the robot is holding block  $x$ ) & (standing next to block  $y$ ) & (the top of  $y$  is clear) THEN (placing  $x$  on  $y$  achieves the state of  $x$  being on top of  $y$ )." The goal matches the conclusion of this rule by replacing  $x$  by  $b$  and  $y$  by  $a$ . The program might then set up as a new goal the achievement of the situation in which the robot was holding block  $b$  and standing next to block  $a$  with the top of block  $a$  clear. That is, the problem containing one goal is replaced by a new problem containing three subgoals, all three of which must be solved in a compatible manner. If there were several rules that matched some part of the current goal (and unfortunately, there usually are), the program would have to decide which one(s) to try and in which order (see below). When some parts of the current goal match a fact (e.g., the starting situation of the problem or a general fact), that part of the goal is said to be solved, and no new subgoals are introduced. Thus, if the initial situation included the fact that block  $a$  had nothing on it, the third of the three subgoals above would already be satisfied. A major difficulty is that solving one subgoal may interfere with the solution of another. For example, the above problem might begin with the robot already next to block  $a$ , so that the first subgoal might appear to have a trivial solution. However, in order to achieve the second subgoal, the robot has to move to where block  $b$  is, undoing the solution to the first goal. Hence the solutions have to be compatible. When all subgoals of some alternative problem are solved, that problem is said to be solved.

In forward reasoning, the specific facts are matched with the conditions of the rules rather than goal parts matching with conclusions. Thus, the program starts with the initial situation rather than the desired end situation. The intermediate steps are situations that can be reached from the starting state rather than secondary goals that must be achieved in order to solve the original problem. When one of these reachable states matches the goal, the problem is solved. As one can imagine, forward chaining tends to generate many more intermediate states than backward chaining for problems in which the goal is fairly well defined because in backward chaining the effort is focused more directly on the given problem. However, there are situations in which forward chaining is more appropriate. There are situations in which the goal is not so easily defined. For example, in chess, the goal is to make a good next move. In such situations forward chaining is used, usually with some kind of evaluation scheme that predicts the relative merit of the intermediate states in achieving the ultimate goal. Another situation where forward chaining is appropriate is the case of consequence finding. For example, in mathematics one may propose a new theory and be interested in seeing sample theorems without knowing the specific propositions to pose or perhaps have a specific example, like a diagram, which focuses the forward inferencing. Another example might be a case in medical diagnosis when a specific drug is being prescribed, and the doctor wants to know if there are any obvious or immediately foreseeable negative side effects. There are many such situations in which there is an explicit, relatively small body of facts from which it is feasible to generate all or most conclusions in a forward manner.

A generalization of the forward-backward control methodology is the set-of-support strategy used mainly in theorem provers. Here the user selects a set of formulas that are to be



given special emphasis. These formulas are said to have support. Initial inferences must use at least one supported formula, and the new conclusions are then also given support. If the selected initial supported formula is the goal formula, the result is backward reasoning; if the selected support consists of the initial-state formulas, forward reasoning results. In many situations a blend of the two methods is effective. For example, it is often beneficial to concentrate on the goal formula and the special facts of the given problem. For example, in mathematical problems these are the conclusion of the theorem to be proved and the special hypotheses (see also Processing, bottom-up and top-down).

**AND/OR Representation of the Search Space.** As pointed out above, at any given point in a backward search the program may have several options for making the next inference step. For example, at the beginning of a backward search there might be five rules whose conclusions match the goal. The program might pick one or more of these and generate a set of subproblems, each one of which might contain several parts. At the next round the program must pick one of these subproblems and one of the subgoals in it and repeat the process. Thus, the program generates a tree of subproblems. At any given level of this tree there is branching on the basis of which rules and/or facts can be used to potentially solve a subgoal within a subproblem. Since any of these rules could potentially solve that subgoal, these branches represent an OR condition among various alternatives. If any alternative works, that subgoal of the subproblem is solved. Each subproblem itself represents a set of subgoals, each of which must be solved in order for the subproblem as a whole to be solved and so represents an AND condition. Thus, such trees are often called AND/OR trees (23) which are pictorially shown in Figure 1.

One speaks of the entire (usually infinite) tree of such possibilities as the search space. A reasoning program physically generates a portion of this tree until it finds some subtree such that for each subproblem in that subtree all the subgoals are solved in a compatible way. This means that looking from the leaves of the tree toward the root, the leaves must all be facts, not rules, which would have introduced new subgoals. The facts and rules in such a subtree provide one way of attaining the goal state from the initial configuration.

Given that the system has chosen a kind of reasoning—forward, backward, or other—generally, there are still a number of additional choices to be made during the search for a solution to a problem posed to the system by a user. Among them are the following important aspects: choosing the next subproblem to work on, choosing a subgoal from that problem,

and choosing one or more rules to apply to that subgoal (see AND/OR graphs).

**Subproblem in the AND/OR Tree on Which to Work.** There are several general techniques for the first of these choices, picking the next subproblem on which to work. Among the earliest methods studied are the so-called blind search methods, depth-first and breadth-first search (23). These correspond exactly to depth-first or breadth-first tree traversal in computer science except that the nodes of the tree are generated during the traversal instead of having the tree already built. Also, suitable provisions must be made for the case when duplicate nodes are generated, i.e., when a particular state can be attained in different ways. They are called “blind” because they make no use of any internal properties of the subproblems and subgoals within those subproblems, only on the position of the subproblem in the expanding tree. Breadth-first search has the interesting property that if a solution will be found, it will be at the lowest level in the tree. Thus, in a situation where a rule might correspond to an operation (e.g., a rule might represent how the world changes when a robot picks up an object) and the operations have equal cost, breadth-first search will find a minimal-cost solution. If the costs of the operations are not all the same, a trivial modification can be made so that the tree is generated according to increasing-cost wave fronts. Finding minimal-cost solutions is often important. For example, the cost of a robot rule might represent how much energy is required to perform the action. It may be desirable for the robot to conserve as much energy as possible. It is important to distinguish between the cost of a solution and the cost of finding it. A reasoning system might actually find a solution more quickly by judiciously exploring the tree as opposed to generating it level by level even though the total cost of the actions in the solution discovered might be higher. Breadth-first search is an example of a complete strategy, i.e., if the problem posed by the user has a solution, breadth-first search will find it unless the system runs out of resources first.

Depth-first search, on the other hand, is not complete. For example, if the system were trying to show how a robot can build a tower and if it were unlucky enough to follow a path in which the robot always moves to the right, the system would never get around to exploring other action sequences. That is, if the system gets stuck going down an infinite path on which there is no solution and since it is using the strategy to follow that path as far as possible, it never will find a solution. Still, in practice, depth-first search combined with other heuristics for choosing which path to explore and which rules to use to solve subgoals has proved effective from the point of view that solutions are obtained to a large percentage of problems that users want solved without exhausting the system's resources.

Another methodology for this choice is ordered search, often called the A\* method (23). In this method a heuristic function  $f(n) = g(n) + h(n)$  is used to select the subproblem to expand at any given point in the search. Here,  $g(n)$  represents the cost of reaching subproblem  $n$  in the tree (i.e., the cost of all the operations that led from the original goal to subproblem  $n$ ) and  $h(n)$  represents an estimate of the amount of effort needed to reach a solution. Keep in mind that these are the costs of applying the sequence of rules leading directly to that state, not the amount of work the program did or may do to find that sequence. For example, in the problem of building a tower,  $h$  could be a count of the blocks left to be stacked plus a measure

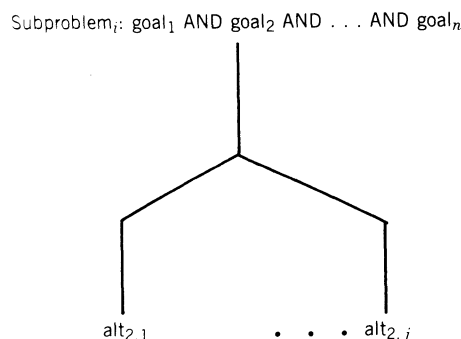


Figure 1. AND/OR trees.

of how far apart those remaining blocks are. In theorem proving,  $h$  could be the number of literals left in a clause. A node with the smallest  $f$  value is always chosen for expansion. If  $h$  is defined to be identically zero, breadth-first search results. Similarly, by setting  $g$  to identically zero and defining  $h$  appropriately, depth-first search can be obtained. In general, by including both the cost  $g$  and the heuristic estimate  $h$ , the system avoids the pitfalls of pure depth-first search while still exploring the tree in some potentially more effective fashion than pure breadth-first order. In addition, if the  $h(n)$  is defined so as to never be higher than the actual cost of reaching a solution from  $n$ ,  $A^*$  is also guaranteed to produce minimal solutions. However, there is still the problem of actually defining  $h(n)$ , and there is no way in general of knowing that the user has satisfied the above requirement. In practice,  $h$  is defined according to the user's or system designer's experience and knowledge about the class of problems to be attacked by the system (see  $A^*$  algorithm; Search entries).

**Which Rule(s) to Apply.** As noted, a particular subgoal in a current subproblem may have several applicable rules, and the system must pick some (subset) of these to apply and pick an order to try them. To date, there exists only a few general techniques by which systems may be directed to make this choice. A great many systems, in fact, make no restriction on applying rules: If there are  $n$  rules that could be used to eliminate a subgoal (or at least replace it by other subgoals), all are allowed to be tried as in the above paragraph. Some systems provide tables of advice as to which rules might be best to use for each of the generic kinds of subgoals and situations. For example, in mathematical reasoning a system might have been programmed to use inverse rules when trying to solve an equality subgoal when one side of the equality is a variable and the other side begins with a function symbol from a special list and not to use any of the general equality axioms. Some interactive systems allow the user to specify which rule to apply to a subgoal. An important related control aspect is which order to try the rules selected for use on a given subgoal. One technique that gets around this question is to apply all of the selected rules at once and then use techniques as described in the above paragraph to select which of the resulting nodes to look at first (or even whether any of them are to be preferred over some other part of the tree generated so far). In some other systems, notably PROLOG, the rules are applied in the order in which they were entered into the system by the user. Thus, some control information is provided just by the order in which the rules occur in memory.

**Which Subgoal to Solve.** Finally, there is the problem of picking the subgoal out of the current subproblem to try to solve. First, it is important that a reasoning program avoid the redundancy inherent in trying to solve all of the subgoals separately or in parallel. A solution to a problem with  $n$  subgoals can be generated in  $n!$  different ways because of the  $n!$  ways in choosing the next subgoal on which to work. Thus, reasoning programs normally try to solve the list of subgoals in a subproblem in some order. As in selecting applicable rules, one possibility is to try to solve the subgoals in the order in which they occur, say, left to right. In such systems the leftmost subgoal is selected, and any new subgoals generated by applying some rule are added at the left, again usually in the same order in which they occur in the rule used. This is used in problems where the order in which subgoals are to be solved is

important to the application of the rules. For example, a rule might have several preconditions and then some additional subproblems to solve before the rule will succeed. The preconditions would then be listed first. Another approach is to provide the system with means to dynamically analyze and reorder the subgoals. In theorem proving, e.g., the literals in a clause are the subproblems. One strategy is to select the literal with the fewest free variables. In general, such systems usually select some subgoal that has the least degree of freedom, the idea being that such a subproblem will have generally fewer applicable rules so that the branching rate in the search space will be smaller. A generalization of this technique is to allow any criteria to be used to reorder the subgoals (e.g., SL resolution in theorem proving, i.e., Linear resolution with Selection function). However, very few selection functions have actually been proposed. Finally, as noted above, solutions to the individual subgoals must be compatible. This can often be a useful side effect of the order in which subgoals are solved. For example, in the tower problem solving the subgoal of holding block  $b$  first leads to a sequence of actions in which previous work in the sequence is not undone by later actions. Again, unfortunately, little is known about how to do this in more than an ad hoc way.

**Knowledge-Representation Schemes for Controlling Inference.** Several proposals have been put forward for information structures that would help control the inferencing process. Examples include frames (13), semantic nets (24), and scripts (25). The idea here is that the data structures themselves can provide guidance to the system as to how the data are to be used. An example of this is the semantic net, which contains nodes representing objects and classes and labeled, directed arcs representing relationships. A sentence like "John took Mary to the theater" would give rise to nodes for John, Mary, and theater in addition to nodes for the class of humans and other concepts. There might be an arc connecting John to Mary labeled with the action as well as an arc connecting each of John and Mary to the node representing the class of humans. There would also, in general, be many other kinds of arcs. A system designed to answer questions might be programmed to follow only certain kinds of arcs in response to certain kinds of questions. The disallowed arcs might lead to valid inferences but not ones the designer feels would be relevant to the given class of questions. Script systems go even further in representing information about events by postulating within the data structure itself prototypes for those events. Going to the theater involves buying tickets, then arriving at a specified time, etc. Foreknowledge of what is expected can guide the reasoning system through the description of a particular event and help the system recognize both particular cases of expected actions and exceptions to the prototype (see Semantic networks; Frame theory; Scripts).

## Conclusion

Reasoning systems involve the representation of information and knowledge, the use of inference rules for manipulating that knowledge and information, and control mechanisms for making the variety of choices necessary in the search for solutions. Each reasoning system will emphasize certain ones of these aspects and implement them in ways appropriate to its own kind of reasoning (see also the following entries on Reasoning).

## BIBLIOGRAPHY

1. A. L. Samuels "Some studies in machine learning using the game of checkers," *IBM J. Res. Dev.* 3, 210-229 (1959).
2. P. C. Gilmore, "A proof method for quantification theory," *IBM J. Res. Dev.* 4, 28-35 (1960).
3. M. Davis and H. Putnam, "A computing procedure for quantification theory," *JACM* 7, 201-215 (1960).
4. A. Newel, J. Shaw, and H. Simon, Empirical Exploration of the Logic Theory Machine, in E. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, pp. 109-133, 1963.
5. J. A. Robinson, "A machine-oriented logic based on the resolution principle," *JACM* 12, 23-41 (1965).
6. L. Wos, D. Carson, and G. Robinson, "Efficiency and completeness of the set-of-support strategy in theorem proving," *JACM* 12, 536-541 (1965).
7. G. Robinson and L. Wos, Paramodulation in First-order Theories with Equality, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 4, American Elsevier, New York, pp. 135-150, 1969.
8. A. Newel and H. Simon, GPS, a Program that Simulates Human Thought, in E. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, pp. 279-293, 1963.
9. Z. Manna, "The correctness of programs," *JCSS* 3, 119-127 (1969).
10. E. Feigenbaum and G. Buchanan, Heuristic DENDRAL: A Program for Generating and Explaining Hypotheses in Organic Chemistry, *Proceedings of the Hawaii International Conference on Systems Sciences*, University of Hawaii Press, Honolulu, 1968.
11. L. Zadeh, "Fuzzy sets," *Inform. Ctrl.* 8, 338-353 (1965).
12. G. Plotkin, A Note on Inductive Generalisation, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 5, Edinburgh University Press, Edinburgh, pp. 153-163, 1970.
13. M. Minsky, A Framework for Representing Knowledge, in P. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.
14. R. Reiter, "A logic for default reasoning," *Artif. Intell.* 13, 81-132 (1980).
15. E. Shortliffe, *Computer-Based Medical Consultations: MYCIN*, Elsevier, New York, 1976.
16. A. Bundy and B. Welham, "Using metalevel inference for selective application of multiple rewrite rules in algebraic manipulation," *Artif. Intell.* 16, 189-211 (1981).
17. H. Gallaire and J. Minker (eds.), *Logic and Databases*, Plenum, New York, 1978.
18. W. Wojciechowski and A. Wojcik, "Automated design of multiple valued logic circuits by automated theorem proving techniques," *IEEE Trans. Comput.* C-32, 785-798 (1983).
19. J. McDermott, R1: A Rule-Based Configurer of Computer systems, Technical Report CMU-CS-80-119, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1980.
20. E. Lusk, W. McCune, and R. Overbeek, Logic Machine Architecture, Kernel Functions, "Proceedings of the Sixth International Conference on Automated Deduction," *Lecture Notes in Computer Science*, Vol. 138, Springer, New York, pp. 70-84, 1984.
21. C. Foggy, The OPS5 User's Manual, Technical Report CMU-CS-81-135, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1979.
22. R. Davis and J. King, "An overview of production systems," in E. Elcock and D. Michie (eds.), *Machine Intelligence*, Wiley, New York, pp. 300-332, 1976.
23. N. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, 1980.
24. N. Findler (ed.), *Associative Networks: Representation and Use of Knowledge in Computers*, Academic Press, New York, 1979.
25. R. Schank and R. Abelson, *Scripts, Plans, Goals and Understanding*, Erlbaum, Hillside, NJ, 1977.

L. HENSCHEN

Northwestern University

## REASONING, CAUSAL

Causal reasoning in AI consists of inferences about the behavior of systems or mechanisms for the purpose of explaining past observations or predicting future events.

Many different types of research in AI fall under this rubric. Although the following categorization captures certain major distinctions in the field, some work fits in more than one category.

Causal links may be asserted between states and events, with inferences to determine the pattern of connecting causal relationships in a particular situation. The pattern is expressed as a causal chain or causal network and may include previously unobserved intermediate states and events.

Symbolic simulation of continuous systems infers the possible behaviors of a continuously changing system from its structure and an initial state. Rather than focusing on states and events, the important objects are continuous state variables, their values at particular times, and the constraints holding among them.

Symbolic simulation of discrete systems also infers the behavior of a mechanism from its structure, but the inference rules for discrete mechanisms, typically digital electronic circuits, are based on the definitions of electronic components rather than on the possible behaviors of continuous functions.

Defining causality and establishing criteria for asserting causal relationships have been the concerns of philosophers of science and statisticians. Formal logicians are also concerned with determining the expressive power a formal logic needs to describe the objects, relationships, and inferences that make up commonsense causal reasoning.

Cognitive mental models provide human examples of representations capable of expressing commonsense knowledge of the physical world. Valuable constraints on the structure of knowledge representations can be derived from the study of child development, human performance on a variety of tasks, and historical, cultural, and individual variation.

## Causal-Link Models

Causal knowledge is often modeled as a network of nodes representing states or events of various kinds, with links representing a variety of causal relationships. These representations have been most extensively explored in two quite different contexts: story understanding (see Story analysis) and medical diagnosis (see Medical advice systems). In both contexts the goal of causal inference is to activate additional nodes as necessary to construct a network of states and events linked in a causally coherent pattern.

Rieger and Grinberg (1) introduced the first extensive causal-link representation with the goal of expressing causal

knowledge of the operation of a complex mechanism sufficient to simulate its behavior. The representation includes nodes for actions, tendencies, states, and state changes and 10 link types representing different causal relationships that might hold among these nodes. The mechanism description consists of local causal relations among nodes, and each node and link is implemented as an independent computational agent. Activation percolates through the network according to the semantics of each type of link. The temporal sequence of node activations is intended to correspond to the sequence of states, actions, and state changes in the behavior of the mechanism. The simulator can, however, introduce spurious temporal ordering relations among events on independent causal paths; no provision is made for detecting or correcting this. Rieger and Grinberg (1) demonstrate that their representation can be applied to a number of nontrivial mechanisms (e.g., the forced hot-air furnace and the flush toilet) and is able to express the structure and simulate the behavior of each mechanism.

Schank and Abelson (2) take a similar approach to causal relationships, making them the basis for an elaborate representation of meaning in natural-language story understanding. Their representation includes nodes (acts and states) and five fundamental link types whose semantics define a simple "causal syntax" of legal causal relationships. These causal relations can then be used to infer the existence of unmentioned intermediate states required for the coherence of a story narrative. For example, "Joe burned his hand because he forgot the stove was on" can be seen to require additional intermediate states and actions to be comprehended; a suitable store of real-world knowledge allows a reasonable causal scenario to be constructed. In Schank and Abelson's (2) overall theory of story understanding, causal chains provide the skeleton for the key concept of the branching script. Stories in which scripts are constructed on the fly can be understood in terms of the actors' plan and goals. The purpose of this application of causal reasoning is to construct a complete and coherent causal scenario constrained by the evidence in the text. Neither the story nor the causal reasoning representation by itself is sufficient to construct an unambiguous scenario, but a well-structured story can be understood by considering both constraints together.

The purpose of causal reasoning in medical diagnosis is similar to its purpose in story understanding: to determine a causally coherent set of events that has taken place and that matches the available observations. The medical-diagnosis systems using causal reasoning typically employ a simpler "causal syntax" consisting of a single type of node representing a partial description of the patient's state and a single type of causal link.

The causal relationships in the CASNET system (3) for diagnosis of glaucoma are probabilistically weighted links between pathophysiological states. A disease process is modeled by a chain of states, where the current point on that chain is the progress of the disease at the current time. Support for diagnostic hypotheses is propagated from clinical evidence to pathophysiological states; among pathophysiological states across causal links, and from chains of pathophysiological states to disease hypotheses. In CASNET causal links reflect real causal relations in the glaucoma domain and therefore can be activated in orderly chains with causally meaningful relations to disease hypotheses. However, the bulk of the inference involving these links and their weights treats them simply as conditional probabilities. Thus, the fact that they are causal relations is relevant to the construction and valida-

tion of the knowledge base but not to the operation of the program.

The ABEL program (4) for reasoning about acid-base and electrolyte disturbances also uses weighted causal links between nodes representing pathophysiological states, but the weights are interpreted as measures of magnitudes rather than as probabilities. This allows ABEL to reason about interactions between multiple diseases by deciding whether a known cause is sufficient to account for the observed magnitude of the disorder. If not, a second disturbance may be interacting with the first to cause the observed problem. ABEL reasons about interactions at a detailed, pathophysiological level of description, where nodes represent the deviations of electrolyte concentrations from their normal values. In order to construct a useful diagnosis, it builds a model of the patient state at several levels of abstraction, linking the pathophysiological description with a description at the clinical level. Although much of its reasoning is symbolic, ABEL relies on numerical calculations to generate disease-interaction hypotheses and to test whether a given cause is sufficient to account for an observed effect. This limits its applicability to domains where adequate numerical theories exist.

In his design for CADUCEUS (qv), the successor to INTERNIST-I as a general diagnostic program for internal medicine, Pople (5) develops a pattern of interaction between causal and taxonomic links among disease hypotheses and pathophysiological states, focusing on the search for appropriate intermediate states. In the causal network part of the representation, nodes represent only states, not events or state changes. There is only one kind of link, meaning "may-be-caused-by," linking a pair of states. In searching for a diagnosis, the causal network proposes differential-diagnosis problems by defining a small set of possible causes for an observed finding or concluded intermediate state. Simultaneously, an independent taxonomic network can provide similar diagnostic problems. Taking advantage of interactions between these two networks, Cadeuceus analyzes the set of proposed differential-diagnosis problems to determine the diagnostic test that best refines the set of hypotheses at lowest cost. Thus, Cadeuceus constructs a network of causally related states in order to focus its search for a good diagnosis and to test candidate diagnostic hypotheses.

In general, causal-link models of causal reasoning assume the existence of a knowledge base of potential states, events, and causal relations. The problem to be solved is to construct a causally coherent subnetwork best corresponding to, and hence explaining, the available observations. The underlying ontology consists entirely of fragments of behavior: events, actions, states, and state changes.

Causal-link models of causal reasoning have been criticized on the grounds that they lack the ability to represent the structure of a mechanism or to derive predictions of behavior from structure. Without this capability, interactions between processes are difficult to reason about since they interact by having simultaneous influences on structural parameters. Only Patil et al. (4) address this issue by assigning numerical strengths to causal links and performing substantial calculations with those numbers.

### Symbolic Simulation of Continuous Systems

Symbolic simulation of a continuous system, often called "qualitative simulation," is a type of causal reasoning in which qualitative descriptions of the possible behaviors of a

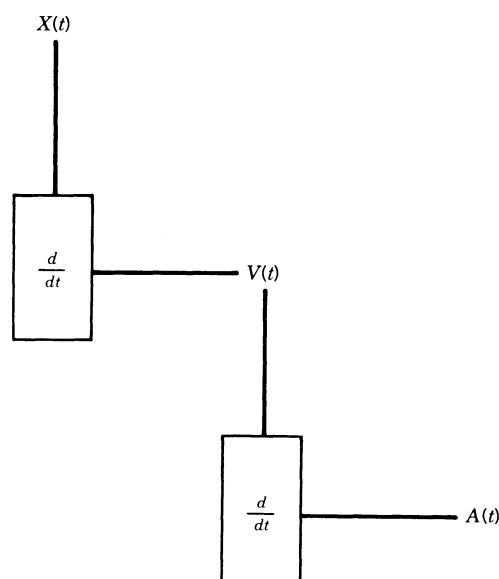
mechanism or system are derived from a qualitative description of its structure. The structural description of a mechanism includes a set of continuous state variables and constraints that must be satisfied by values of those variables at each point in time. In the behavioral description the value of a variable is described qualitatively as a point or interval in a (possibly partially ordered) quantity space, along with its direction of change. A single behavior of a mechanism is described as the sequence of qualitative states of the set of state variables. Starting with a qualitative structural description, there may not be enough information to specify the behavior uniquely, so qualitative simulation may predict several possible behaviors. This level of structural description, stated in terms of state variables and the constraints on their values, can be considered a qualitative abstraction of a differential equation. The behaviors derived by qualitative simulation are intended as qualitative abstractions of the solutions to that differential equation (6). Figure 1 gives an example of qualitative structure and behavior descriptions.

Different researchers propose different means for acquiring this structural description. Kuipers and Kassirer (7), working primarily in a medical domain, assume that the structural description is given in the knowledge base rather than derived from the physical anatomy of the mechanism. DeKleer (8,9) and Williams (10,11) derive structural constraints from physical anatomy: the components and circuit topology of analog electrical circuits. Forbus (12,13), studying commonsense reasoning about physical situations, examines the physical anatomy, along with what is known about the quantities involved, to determine a set of active processes. The structural description is then derived from the set of processes. The process-based approach uses the concept of spatially and temporally localized histories (14) to infer restrictions on the set of possible causal interactions in the situations. The other approaches assume a closed set of possible interactions specified by the structural description or circuit diagram.

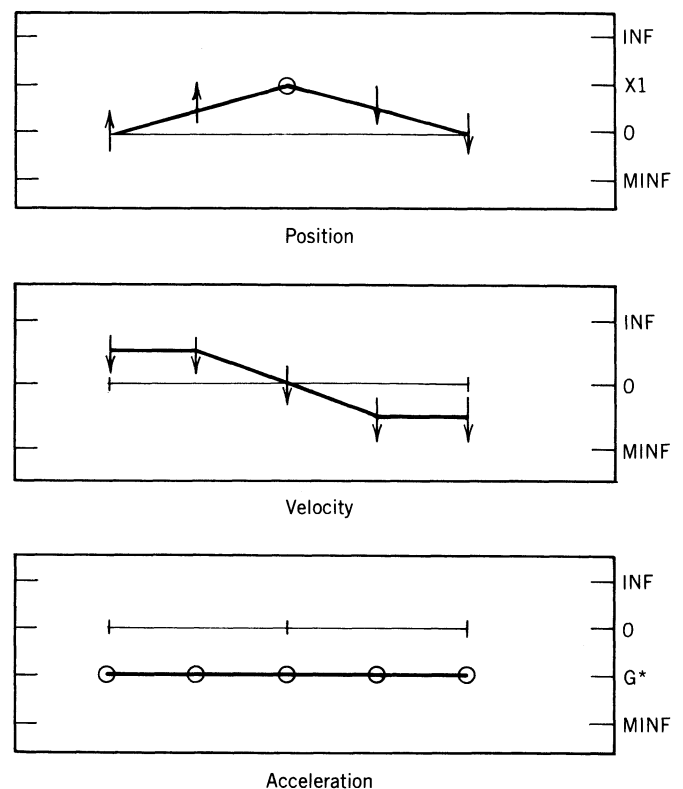
When the structural description is constructed from the physical anatomy of the device, components are described in terms of their behavior. The no-function-in-structure principle (9) is an important restriction on the behavior description of components: A component may be described in terms of its context-independent behavioral repertoire but not in terms that presume any properties of the mechanism in which it is embedded. Thus, the component description of a switch may not presume the existence of a current that will flow when the switch is closed and not when it is open. Rather, the switch might be described as having very low resistance when closed and very high when open, leaving the rest of its description (i.e., current and voltages) to be filled in by the prevailing context of a particular mechanism.

There are important similarities and differences among the different qualitative representations and simulation algorithms. Given a structural description of a mechanism in terms of state variables and constraints, descriptions of some of the variable values can be propagated across the constraints to create a more complete description of the mechanism's state at a given point in time. Once the current state is adequately described, those variables with changing values are examined to determine if they are approaching limiting landmark values. The next qualitatively distinct state is determined by analyzing the possible transitions of individual variables to determine which transitions can occur next using either a transition ordering decision procedure (10,13,15) or constraint-based transition filtering rules (6,9). A given qualitative state of a mechanism may have zero or more successors, so qualitative simulation creates a branching tree or directed graph of state descriptions as the description of behavior. Other similarities and differences include those described in the next three paragraphs.

Values are described qualitatively in terms of their ordinal relations with a small set of distinguished "landmark" values. The value description also specifies the current direction of



**Figure 1.** Structural and behavioral descriptions of a ball rising and then falling in constant gravity. The structural description gives the derivative relations between  $X(t)$ ,  $V(t)$ . In the behavioral description both time and quantity axes are qualitative: Values can be described as being either equal to, or in the interval between, landmark values.



change of the variable. Some investigators allow zero as the only landmark (8,11), some allow new landmarks to be discovered (6,15), and some allow landmarks to be partially ordered (13).

Forbus's (12,13) influences are open-ended relations stating how changes to one variable affects another, all else being equal. Only the complete set of influences can be analyzed to determine how the variables will co-vary. For the other investigators the structural description consists of constraints, each of which must always be satisfied.

De Kleer (8,9) uses an unusual style of simulation called envisionment, which constructs all the combinatorially possible qualitative states of the system and then links them into a transition network, capturing all possible behaviors of the mechanism. The other investigators simulate forward in time from an initial state.

The qualitative simulation approach to reasoning about mechanisms does not make use of explicit causal relations in its inference process. Rather, it starts with physical anatomy, extracts a structural description, and derives a set of possible behaviors. In some cases assertions of causal relationships among events are derived from the order of events in the behavioral description and the order of information propagation during simulation. Clearly, qualitative simulation does not explicate the full concept of causality in mechanisms. On the other hand, the use of qualitative simulation to derive predictions of the behavior of mechanisms is a useful method for reasoning about the physical world.

### Symbolic Simulation of Discrete Systems

In a continuous system symbolic simulation involves mapping a continuous space of values into a discrete space of qualitative descriptions. Simulation algorithms are based on the properties of continuously differentiable functions: the intermediate-value and mean-value theorems. In a digital system, on the other hand, state variables already have discrete values. The set of permissible state changes is not constrained by continuity but by the semantics of the components in the particular circuit. Thus, rather than having a small closed set of qualitative state-transition rules, a digital simulator must have an open-ended catalog of device descriptions.

Although symbolic descriptions of variable values are not used to represent open intervals of values, they are needed to handle abstractions of values or disjunctions of possible values when sufficient information is not available to specify the value uniquely. In the case of automatic verification of the design of digital circuits (16), symbolic descriptions allow the specifications for a module to be used both as the overall behavioral description to be justified from the module's internal structure and as the context-independent component description when the module is used in a larger structure. Using this modular, hierarchical decomposition, Barrow (16) shows how a substantial, realistic, digital circuit can be effectively verified by simulating its behavior and comparing the predicted behavior with its specification. Verification of the lower level modules produces stored "lemmas" that can be referred to later in the proof, a strategy that greatly shortens the overall reasoning time. The decomposition also allows different levels of the hierarchy to operate in very different value domains: voltage levels, logic values, integers, etc. The success of this approach depends strongly on the modular decomposition: a reasonable assumption in the case of designed structures such

as digital circuits but less reasonable in, say, biology. The clarity of the hierarchical abstraction relations also stems from the fact that both structural and behavioral descriptions for modules can be expressed as equations relating the values of input, output, and state variables.

When the behavior of a digital circuit fails to meet its specifications, the problem is frequently not that the design was incorrect but that the actual circuit fails to satisfy the structural description in the design. A hypothesized fault in the circuit, once suspected, can be tested by simulating the faulty circuit forward from a given set of test inputs to see whether the prediction matches the observed outputs. Thus, simulation of circuits under fault hypotheses is used as the test portion of a generate-and-test diagnostic search (qv). Troubleshooting is difficult because the generation of fault hypotheses is highly underconstrained, and there is an enormous space of possibilities. Davis (17) describes a hardware troubleshooting system whose generation of fault hypotheses is controlled by a set of assumptions about the nature of the fault, which are progressively suspended as the search goes on so the most likely portion of the space is considered first. Davis also uses multiple concepts of structural adjacency—e.g., topological, physical, thermal, and electromagnetic—to generate fault hypotheses that would have been invisible from other vantage points. For example, the set of possible bridge faults inappropriately connecting two wires in a circuit can be difficult to express on the circuit schematic representation but are nonetheless tightly constrained to occur only between physically adjacent wires on the circuit board. Thus, a description capturing spatial adjacency must be available to generate the appropriate set of hypotheses.

Causal reasoning and symbolic digital simulation are key elements, but still only a small part, of a large domain of literature on digital-circuit verification, test generation, and automated troubleshooting. Symbolic simulation of digital circuits is feasible for the same reasons that such circuits are useful in computers: both time and the state variables can be treated as having discrete values, and the state variables can be interpreted simultaneously as lying in more than one semantically distinct space of values. Thus, causal reasoning about digital circuits is a specialized application with specialized assumptions on which it can draw. Although this line of research is likely to produce useful and impressive results, those results are not strongly linked with the study of causal reasoning as part of commonsense knowledge about the everyday physical world.

### Defining Causality

Some of the work described above takes as primitive the notion that one event can cause another and derives implications from the relationship without defining the meaning of "causality." Other research derives behavior from structure using symbolic simulation without any essential reference to causality at all. The meaning of causality and the criteria for legitimately asserting a causal relationship given observations of regularities in the world have been of concern to philosophers of science and statisticians.

In his role as philosopher of science, Simon (18) defines the causal relation to hold between sentences describing the world rather than between events or states in the world. His definition provides criteria for asserting that one sentence causally precedes another with respect to a given set of laws describing



the world. The criteria are intended to model the ordinary usage of the term "causality," including the irreversibility of the causal relations, so that it can be appropriate to say "The rain causes John to wear his raincoat" but not the contrapositive "John's not wearing his raincoat causes it not to rain." Simon's (18) approach to causality is, of course, one position on a major philosophical issue that extends far beyond the scope of this entry.

In the RX project Blum (19) develops an "automated statistician" that examines medical evidence stored in a time-oriented database and applies the appropriate statistical tests to determine whether a causal relationship can be inferred. RX follows accepted procedures in statistical data analysis and uses an operational definition of causality: A is said to cause B if, over repeated observations, A is generally followed by B, the intensity of A is correlated with the intensity of B, and there is no known third variable C responsible for the correlation. Similar to Ref. 4, causal links in RX have a number of attributes including numerical ones such as intensity and frequency. The third test of causality, for nonspuriousness, is the central concern of RX and is responsible for the bulk of its domain-specific knowledge base of known causal links.

Knowledge of the commonsense physical world, however, is much more complex and includes much knowledge that must be considered "causal" but that does not fit neatly into the causal link or symbolic simulation categories of representation. For example, we know that water poured from a canteen might splash off a rock, wetting both the rock and one's legs, and then soak into the ground. The main thrust of the research described in previous sections is to create the inferential power to understand or predict the behavior of complex mechanisms. However, examples such as the pouring water demonstrate that a major unsolved problem in causal reasoning is the expressive power to state, adequately, the simple facts and simple relationships that constitute commonsense knowledge about the physical world. Hayes (14) argues that causality, as such, is not a self-contained category that can support independent axiomatization. Rather, it is a type of knowledge that needs to be represented separately for different physical domains: causal knowledge about liquids is knowledge about liquids, not about causality.

In "The Naive Physics Manifesto" (see Physics, naive) Hayes (14) argues for a research program to develop representation languages with sufficient expressive power to handle issues such as these. He lays out a set of domains in which progress would be possible and useful: scales for measurement; shape, orientation, and dimension; inside and outside; histories; energy and effort; assemblies; support; substances and physical state; forces and movement; and liquids. The concept of histories, particularly, has been useful for constraining the types of causal interactions that need to be considered in a given situation and thus provides a way to avoid the frame problem (compare Refs. 12 and 13). An object or event is considered to have four-dimensional spatiotemporal extent, called its history. One's knowledge of an object or event includes knowledge of how far it is likely to be extended and what kinds of boundaries define its limits in both space and time. Two objects can interact only if their histories intersect. Thus, moving a block on a tabletop cannot affect a block that is now in the other room because the histories of the two blocks do not intersect and because the history of a move action does not have a sufficient radius to affect both blocks. On the other hand, the move action could potentially affect other blocks on

the same table so a more careful check for collisions is warranted.

McDermott (20) proposes a semantic foundation for the concept of time, which includes treatments of events, causal relations, continuous change of quantities, and the persistence of facts. In this theory time is organized as a branching set of chronicles, with a single past and multiple possible futures. A chronicle is a global history of the entire universe, in contrast to the spatially bounded histories belonging to individual objects in Hayes's (14) approach. In order to handle the frame problem, McDermott associates with each fact a persistence or default period of time during which that fact, once known, can be assumed to remain true in the absence of information to the contrary. With these as foundation, his logic makes it possible to assert causal relations among events in the causal-link style, to define the set of influences on a continuously changing quantity in the qualitative-simulation style, and to plan or predict future events. The device of alternate future chronicles allows him to handle some cases of the semantically difficult verb "prevent," referring to the relation between a current action and a future event that does not take place.

In this very active research area certain influential papers such as Hayes's "Ontology for Liquids" and the fruits of the "Commonsense Summer" project circulate only in prepublication form. These and similar foundational studies are essential to the eventual integration of different and seemingly incompatible approaches to causal reasoning into a common framework.

### Cognitive Mental Models

The search for new knowledge representations (qv) in AI benefits by considering the constraints on human representations discovered by cognitive scientists (see Cognitive science). In order to describe human knowledge, a representation must be able to express the kinds of variation—individual, developmental, and historical—in knowledge and problem-solving performance that is observed in the human population. Evidence of this expressive power suggests design directions for knowledge representations.

As in many other domains of knowledge, Piaget (21) provides the earliest systematic studies of causal knowledge, looking at the types of early theories children create to explain such mechanisms as the wind blowing, clouds moving, boats floating, water levels, shadows, bicycles, steam engines, trains, motorcars, and airplanes. In Ref. 21 he identifies a large variety of distinct types of causal explanation grouped into three classes: the psychological/magical, the animistic, and finally the rational/mechanical. Only the third class encompasses what we normally consider as "causality." The evolution through these classes is driven by three processes of cognitive development: the progression from subjective to an objective view of causality as external to the self, from the view of causality as an almost immediate co-occurrence of events to an ordered sequence of intermediate steps, and from causal relations seen as irreversible changes to reversible mechanical connections.

Causal knowledge has great influence on other kinds of reasoning. Tversky and Kahneman (22) have shown the importance of causal knowledge in the overall problem-solving behavior of people by demonstrating the strong dominance of schemas of causal relations over the phenomena that influence estimates of relative and absolute probabilities.

Adults frequently have incorrect models of physical causality. McCloskey, Caramazza, and Green (23) demonstrate a number of consistent errors in adult predictions for physical situations involving curvilinear motion. Many of their subjects behaved as though curvilinear motion were not the result of an imposed force but as if it were a property of motion like velocity that could change only gradually under external influences. For example, a pea blown out of a spiral peashooter is predicted to follow a gradually flattening curve rather than a straight line (because of "conservation of angular momentum"). These and similar experimental examples bear a striking resemblance to the pre-Galilean "impetus" theory of motion (24,25), suggesting that the history of science may contain useful evidence about the nature and evolution of knowledge representations. Wise and Carey (26), however, caution that scientific theories of an earlier age were embedded in elaborate belief systems that differ greatly from current theories, so structural analogies between the history of science and commonsense knowledge in specific problem domains should be treated with care.

Constraints from empirical observations of human behavior may be combined with the computational constraints of the problem to discover or evaluate novel knowledge representations. Williams, Hollan, and Stevens (27) demonstrate this strategy by collecting and analyzing protocols of subjects reasoning about a heat exchanger and creating a set of different computational models corresponding to the different models observed in the subjects. Similarly, Kuipers and Kassirer (7) collect and analyze a protocol of an expert physician explaining a physiological mechanism and combine empirical and computational constraints to justify a qualitative simulation representation.

## BIBLIOGRAPHY

1. C. Rieger and M. Grinberg, The Declarative Representation and Procedural Simulation of Causality in Physical Mechanisms, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, pp. 250–256, 1977.
2. R. C. Schank and R. P. Abelson, *Scripts, Plans, Goals and Understanding*, Erlbaum, Hillsdale, NJ, 1977.
3. S. M. Weiss, C. A. Kulikowski, S. Amarel, and A. Safir, "A model-based method for computer-aided medical decision-making," *Artif. Intell.* **11**, 145–172 (1978).
4. R. S. Patil, P. Szolovits, and W. B. Schwartz, Causal Understanding of Patient Illness in Medical Diagnosis, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC, pp. 893–899, 1981.
5. H. E. Pople, Jr., Heuristic Methods for Imposing Structure on Ill Structured Problems: The Structuring of Medical Diagnostics, in P. Szolovits (ed.), *Artificial Intelligence in Medicine*, AAAS/Westview, Boulder, CO, pp. 119–190.
6. B. J. Kuipers, The Limits of Qualitative Simulation, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 128–136, 1985.
7. B. J. Kuipers and J. P. Kassirer, "Causal reasoning in medicine: Analysis of a protocol," *Cog. Sci.* **8**, 363–385 (1984).
8. J. de Kleer, The Origin and Resolution of Ambiguities in Causal Arguments, *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, pp. 197–203, 1979.
9. J. de Kleer and J. S. Brown, "A qualitative physics based on confluences," *Artif. Intell.* **24**, 7–83 (1984).
10. B. Williams, The Use of Continuity in a Qualitative Physics, *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, pp. 350–354, 1984.
11. B. Williams, "Qualitative analysis of MOS circuits," *Artif. Intell.* **24**, 281–346 (1984).
12. K. D. Forbus, Qualitative Reasoning about Physical Processes, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC, pp. 326–330, 1981.
13. K. D. Forbus, "Qualitative process theory," *Artif. Intell.* **24**, 85–168 (1984).
14. P. J. Hayes, The Naive Physics Manifesto, in D. Michie (ed.), *Expert Systems in the Micro-Electronic Age*, Edinburgh University Press, Edinburgh, pp. 242–270, 1979.
15. B. J. Kuipers, "Commonsense reasoning about causality: Deriving behavior from structure," *Artif. Intell.* **24**, 169–204 (1984).
16. H. G. Barrow, "VERIFY: A program for proving correctness of digital hardware designs," *Artif. Intell.* **24**, 437–491 (1984).
17. R. Davis, "Diagnostic reasoning based on structure and behavior," *Artif. Intell.* **24**, 347–410 (1984).
18. H. A. Simon, *Models of Discovery*, D. Reidel, Boston, 1977.
19. R. L. Blum, *Discovery and Representation of Causal Relationships from a Large Time-Oriented Clinical Database: The RX Project*, Lecture Notes in Medical Informatics, Vol. 19. Springer, New York, p. 19, 1982.
20. D. McDermott, "A temporal logic for reasoning about processes and plans," *Cog. Sci.* **6**, 101–155 (1982).
21. J. Piaget, *The Child's Conception of Physical Causality*, Routledge and Kegan Paul, London, 1930. Reprinted by Littlefield, Adams & Co, Totowa, NJ, 1966.
22. A. Tversky and D. Kahneman, Causal Schemas in Judgments under Uncertainty, in M. Fishbein (ed.), *Progress in Social Psychology*, Vol. 1. Erlbaum, Hillsdale, NJ, 1980.
23. M. McCloskey, A Caramazza, and B. Green, "Curvilinear motion in the absence of external forces: Naive beliefs about the motion of objects," *Science* **210**, 1139–1141 (1980).
24. M. McCloskey, Naive Theories of Motion, in D. Gentner and A. Stevens (eds.), *Mental Models*, Erlbaum, Hillsdale, NJ, pp. 299–324, 1983.
25. J. Clement, A Conceptual Model Discussed by Galileo and Used Intuitively by Physics Students, in D. Gentner and A. Stevens (eds.), *Mental Models*, Erlbaum, Hillsdale, NJ, pp. 325–340, 1983.
26. M. Wiser and S. Carey, When Heat and Temperature Were One, in D. Gentner and A. Stevens (eds.), *Mental Models*. Erlbaum, Hillsdale, NJ, pp. 267–298, 1983.
27. M. D. Williams, J. D. Hollan, and A. L. Stevens, Human Reasoning about a Simple Physical System, in D. Gentner and A. Stevens (eds.), *Mental Models*, Erlbaum, Hillsdale, NJ, pp. 131–154, 1983.
28. D. G. Bobrow (ed.), *Qualitative Reasoning about Physical Systems*, Books/MIT Press, Cambridge, MA, 1985. A collection of papers, reprinting the contents of *Artificial Intelligence* **24** (1984) and giving an excellent overview of the qualitative simulation and symbolic digital simulation types of causal reasoning.
29. D. Gentner and A. Stevens (eds.), *Mental Models*, Erlbaum, Hillsdale, NJ, 1983. A collection of papers giving the best overview of cognitive science research on causal reasoning.

## REASONING, COMMONSENSE

For an artificial system to act sensibly in the real world, it must know about that world, and it must be able to use its knowledge effectively. The common knowledge about the world that is possessed by every schoolchild and the methods for making obvious inferences from this knowledge are called common sense in both humans and computers. Almost every type of intelligent task—natural-language processing (qv), planning (qv), learning (qv), high-level vision (qv), expert-level reasoning (qv)—requires some degree of commonsense reasoning to carry out. The construction of a program with common sense is arguably the central problem in AI.

It is, however, a very difficult problem. Common sense involves many subtle modes of reasoning and a vast body of knowledge with complex interactions. The analysis of commonsense knowledge leads into many deep philosophical problems. Consider the following quotation from *The Tale of Benjamin Bunny*, by Beatrix Potter:

*Peter did not eat anything; he said he should like to go home. Presently he dropped half the onions.*

Except that Peter is a rabbit, there is nothing subtle or strange here, and the passage is easily understood by five-year-old children. Yet these three clauses involve, implicitly or explicitly, concepts of quantity, space, time, physics, goals, plans, needs, and speech acts. An intelligent system cannot, therefore, understand this passage unless it possesses a theory of each of these domains, a collection of relevant facts, and the ability to connect these facts in a useful way to the story. The goal of research in commonsense reasoning is, thus, to create such theories, to state the prominent facts in them, to describe how such knowledge is used, and to implement all this in a working computer program.

### Methodology

A theory of commonsense reasoning in a particular domain consists of three major parts: a theory of the domain, a representation of knowledge, and a method of inference. The domain theory consists of an ontology that systematically describes the domain and an account of what types of problems are likely to arise and what kinds of inferences likely to be needed in reasoning about the domain. A representation consists of a formal language, which can be used to state facts in the domain, and an implementation, which describes how a set of facts in the language can be represented as a data structure. The theory of inference consists of a description of inferential procedures that operate on the implementation; an abstract description of the inferential processes in terms of statements in the language; and a justification in terms of the domain theory of why such inferences are sensible and why they will run in reasonable time.

Two major methodological issues separate researchers in commonsense reasoning. The first question is to what extent the constructs of the formal language should correspond to the constructs of a natural language. The second question is whether the theory should be based on the predicate calculus: whether the formal language should be a first-order language in predicate calculus; whether it should be related to the ontology through a Tarskian semantics; whether the inference pro-

cedures should be sound deductively or given some other formal justification; and whether inference should be carried out using general-purpose deductive techniques (1–9). The two questions are orthogonal: For example, representations based on Montague semantics combine logical with natural-language constructs (10); Wilks uses natural-language constructs but avoids formal definitions (6); Hayes uses a strict logic to define his terms but does not follow natural-language constructs (3); and Schank uses neither logical definitions nor natural-language constructs (11). The choice of methodology often depends on domain. Researchers in complex, poorly structured domains tend to use natural-language terms, since they are available, and to avoid logical definitions, since they are hard to formulate. Researchers in simple, highly structured domains prefer to avoid natural language, which is often unnecessarily convoluted, and to use logical definitions, since precision is desirable and relatively easy to attain. The current mainstream opinion favors theories that have a formal structure and that are not based on natural language, but both questions are still open. (7)

To some extent, these issues are reflections of similar issues in the philosophy of language and epistemology (qv). The view that theories should be formal and language-independent derives ultimately from the example of mathematics and was argued by Russell and related philosophers, particularly by the logical positivists. The view that theories should be informal and based on natural language is related to the “ordinary-language” philosophy of Austin and Wittgenstein. The view that natural language can be associated with a logical structure is particularly associated with Montague. There does not seem to be any philosophical school corresponding to the view that representations should be connected to neither logic nor language; it is difficult to see what such a philosophical position would be. The appeal of this view in AI lies in its simplicity from a pragmatic standpoint.

A formal language in a theory of commonsense reasoning must satisfy three requirements. First, the language must be able to describe the relevant aspects of the domain involved. Second, it must be possible to use the language to express the kinds of partial knowledge typically available; the design of the language must take account of likely kinds of ignorance. Third, it must be possible to implement useful inferences as computations using statements in the language. These requirements are called ontological adequacy; expressivity, or epistemic adequacy; and effectiveness, or heuristic adequacy (12). Each of these requirements is defined relative to a certain set of problems; a representation may be adequate for one kind of problem but not for another.

### General Theories of Inference

Some of the inferential techniques used in commonsense reasoning are peculiar to a particular domain; these are discussed below under the separate domains. Often, however, it is possible to use techniques of more general applications. One common general technique for performing inferences that are logically sound—i.e., inferences in which the conclusion follows necessarily from the premises—is resolution. Resolution (qv) is demonstrably complete, in the sense that any logically necessary conclusion can be found using it. In its most general form, however, it is known to be very inefficient. This inefficiency can be mitigated either by using more powerful theo-

rem-proving (qv) techniques or by including with each fact in the system an indication of how that fact should be used (13,14).

One data structure used for knowledge representation (qv), which supports efficient inference of a limited kind, is the associative network, also called the semantic net (qv). In its simplest form a semantic net is a directed graph whose nodes represent individuals or sets and whose arcs represent binary relations. The relationships of membership and containment are distinguished and are treated specially by the inference procedure. Thus, a simple semantic net is capable of expressing such facts as "All polar bears live in the Arctic," "Peter is a polar bear," "Any polar bear is larger than any salmon," and "All polar bears are mammals." A simple inference procedure can be built capable of inferring such facts as "Peter lives in the Arctic" or "Peter is a mammal." Such inferences can be done efficiently using graph search. Semantic nets can be made to express facts of greater logical complexity, but only at the cost of making the inferential system more complicated and slower (15–17).

Many commonsense deductions are not logically necessary, merely plausible. In reading the above quotation from *Benjamin Bunny*, e.g., it is natural to conclude that Peter Rabbit wanted to go home, even though the text says only that he said he wanted to go home, and he might have been lying. The major problems with plausible inferences (see Reasoning, plausible) are how they can be characterized, how they can be withdrawn if they are found to be false, and how degrees of plausibility can be compared. The first two problems lead to the construction of nonmonotonic logics, in which the inference of a conclusion may rest on the absence of controverting evidence. A number of systems of nonmonotonic logic (see Reasoning, nonmonotonic) have been proposed, none of them wholly successful (18–20). There has been better success at implementing such inference techniques in "reason-maintenance" programs that can make and withdraw provisional assumptions (21–23) (see also Belief revision).

The problem of comparing plausibility of conclusions is generally addressed by associating numbers measuring plausibility with each statement and each rule. The issues in such systems are "What do the numbers mean?" "How are the numbers combined when inferences are performed?" and "Where do the numbers come from?" Various different systems of plausibility measurement have been studied, including Bayesian probability (24,25), fuzzy set theory (26), and evidence theory (27), answering these questions in different ways.

### Frame Theory

In many commonsense domains the basic concepts correspond to large systems of relations that are instantiated in many separate instances in the world. For example, the concept of a house involves both physical relationships between many different physical objects (walls, doors, windows) and functional relations, such as being lived in. Each individual house is an instance of this concept and satisfies these relations. The concept of a visit to the doctor involves a structure of events and scenes (waiting in the waiting room, seeing the nurse, etc.) The concept of motherhood is a complex of events, emotions, and social relations. Such concepts are called frames or schematas (1) (see Frame theory). Typically, frames are used by first using characteristics facts to recognize an instance of the

frame and then using the frame to predict the truth of more facts.

A frame is represented as a collection of slots, which are characteristic variables, and relations on these slots. If the components of a particular entity are found to satisfy a sufficient number of these relationships, the entity can be identified as an instance of the frame. The frame is then instantiated, by associating each relevant component (called a filler) with the corresponding slot. It is then possible to infer the remainder of the relationships on the entity and to look for fillers of the remaining slots as further components of the entities. For example, once a door, a wall, and a number of windows have been detected in an image, the "house" frame may be instantiated and the perceived parts of the house associated with the expected components of the house frame. The frame can then be used to interpret further parts of the image or to make predictions about the house. For instance, one may now interpret a few bricks seen above the roof as a chimney, or one may make a guess as to the distance the house extends in back. Though the basic theory is straightforward and easily implemented in simple cases, the logical analysis of frames and their implementation in complex cases is quite problematic (28–31).

### Domain Models

In the remainder of this entry some important aspects of various commonsense domains are discussed: quantities, time, space, physics, intelligent agents, and interpersonal interactions. The categorization is obviously somewhat arbitrary, and many AI theories span several categories.

**Quantities.** Quantities—parameters with magnitudes—are ubiquitous in commonsense reasoning. Spatial reasoning (qv) involves quantities such as lengths and angles; temporal reasoning (qv) involves dates and durations; physical reasoning (see Physics, naive) involves a whole host of quantities, such as mass, energy, temperature, etc.; even naive psychology uses such concepts as degree of happiness or degree of belief (see Emotion modeling). Since reasoning about quantities often involves specialized techniques, AI programs often use a special-purpose module, called a quantity knowledge base, to maintain the facts about quantities known to the system and to perform inferences on them.

One might suppose that representing facts about quantities and using them for inference was simply a standard mathematical problem. However, though mathematical analysis is invaluable in devising and evaluating a quantity knowledge base, nonetheless AI applications have their own peculiar requirements and standards that are somewhat at variance with standard mathematical objectives. In general, AI systems try to trade away accuracy and completeness in favor of robustness under limitations of time and information. Commonsense reasoning generally demands only a rough, qualitative idea of the properties of a system. If these properties depend in a very delicate way on exact quantity values, the problem is generally more appropriate for a specialized system than a commonsense reasoner. On the other hand, AI systems must often give reasonable answers in real time, or limited time, on the basis of extremely partial and vague beliefs.

Generally, quantities are taken to be integer-valued or real-valued, though other ontologies, such as tolerance spaces

(3), interval spaces (32), and the nonstandard real line with infinitesimals (33,34), have been proposed and employed. The formal language on quantities used in any particular application generally consists of a small number of standard mathematical functions and relations. For example, it has been found that in some cases physical situations can be adequately described in terms of "qualitative differential equations," relations that involve only order relations, relevant physical constants, the differential operator, and Boolean operators (33-36). Temporal reasoning generally involves only linear relationships between times (37). Spatial reasoning, on the other hand, in general requires a more complex vocabulary including the distance function, linear mappings, and trigonometric functions (38,39).

In each domain the task of a quantitative knowledge base is to answer queries or to perform inferences given a collection of facts relating quantities. Each application of quantities has its own characteristics in terms of the mathematical nature of the facts and queries involved and the requirements in speed, completeness, and accuracy. The determination of these characteristics and the construction of computational techniques tailored to them is an important research question, which, as yet, has barely been addressed.

**Space.** Spatial reasoning (qv) deals with the positions, shapes, and motions of physical objects and other spatially limited entities. Planning (qv), robotics (qv), physical reasoning, and high-level vision (qv) all require many different kinds of spatial knowledge and inference. Typical problems include map maintenance, keeping track of the known positions of objects and inferring spatial relations (38-40); motion extrapolation, predicting future spatial relations; shape and scene identification, determining whether a perceived object or scene corresponds to a known shape or scene description (41-43); and special-purpose inferences, including physically relevant inferences, such as whether a cup will hold water.

The ontology of spatial reasoning presents few problems. Physical space is taken to be two- or three-dimensional space. (An exception is Ref. 40, which treats space as a graph of point locations, and the routes between them.) At any fixed time an object occupies some connected subset of the space.

Finding a language for spatial knowledge that is expressive, computationally tractable, and well-defined is very difficult. The most difficult issue is shape description. Many different schemes have been suggested to handle this problem, none of them fully adequate. The following are some of the more popular three-dimensional representations (for extensive surveys, see Refs. 43 and 44):

*Constructive solid geometry.* A shape is characterized as the union and difference of a small class of primitive shapes.

*Generalized cylinders.* A generalized cylinder (qv) is the volume swept out when a given two-dimensional cross section is moved along a given axis.

*Oct-trees.* An oct-tree divides space into cells and labels these cells as either entirely outside the shape, entirely inside it, or partially overlapping the shape. Partially overlapping cells may be more precisely described by subdividing them; this subdivision process creates a tree structure.

*Polyhedra.*

*Quadrics.* Portions of the boundary of a shape are described by second-degree equations.

*Extended Gaussian image.* A convex shape is described by listing, for each normal to the surface, the direction of the normal and the area (in some cases, a differential area) of the portion of the boundary that has that normal.

Any such system of shape descriptions can only give exact descriptions for a limited class of shapes. For example, if shapes are represented by polyhedra, shapes with smooth surfaces cannot be represented exactly. Therefore, such a system must have associated with it a measure of approximation that states what class of actual shapes can be approximated by a given shape description. Inference procedures must be sensitive to these approximation measures and perform only inferences that are valid for the whole class of shapes that can be correctly approximated by the particular description.

Positions are described either using topological relationships, such as "inside," "overlaps," and "interlocks," or by constraining the relative positions of points in the objects. In the latter case it may be desirable either to constrain the relative position of two distinguished points, as in "The center of A is to the left of the center of B," or to quantify over all points, as in "All of A is lower than all of B." The position representation must be designed to interact smoothly with the shape representation, both because representations for complex shapes are often constructed by specifying the positions of component parts and because it is often necessary to specify the relative positions of parts of different objects, and these parts must be described in terms of the shape representation. For instance, one might wish to say, "The north side of the library is 500 feet from the south side of the park," or "The 'toe' of Italy points toward Sicily."

**Time.** Time (see Reasoning, temporal) and change are among the most important and difficult issues in commonsense reasoning. They enter into practically every application; very few problems can be construed in purely static terms. However, there are real difficulties in constructing a system that can reason about time robustly.

The major issues in reasoning about time are straightforward. The state of the world changes over time. Events occur over time. Events and states are related by causal laws. Time has a quantitative aspect; nonrelativistically, both instants-of-time and lengths-of-time intervals can be taken as real-valued quantities. Time is continuous, and some events involve continuous change in parameters.

A number of different ontologies for time have been studied. In the situation calculus (12,45) each instant of time is associated with a particular state of the world as a whole, called a situation. An event is a transition from one situation to another; for instance, an event of the block falling onto the ground goes from a situation in which the block is on the table to one where it is on the ground. A time line is thus a sequence of situations connected by events. McDermott (46) modifies this by proposing that situations form a continuous series and that they branch into the future, where branch points represent choices by an agent. Events in this model are associated with intervals of time; for instance, an event of waiting is represented by an assertion about the interval when it takes place. Note that such an event cannot be well represented in the situation calculus since there is no change of state necessarily associated. Allen (47) further modifies McDermott's ontology by excluding situations altogether and using only intervals as fundamental units of time. Hayes (48) approaches the

problem from a different direction and carves up space-time into four-dimensional chunks called "histories." Causal laws are then stated by specifying allowable structures on histories and constraints for piecing histories together.

Each ontology necessarily has a different formal language associated. In general, however, the three kinds of relations needed may be distinguished: relations between temporal entities, (situations, intervals, or histories), such as "Situation *S* was 10 minutes before situation *T*"; relations specifying the states and events associated with a temporal entity, such as "The dog was barking during interval *I*"; and causal relationships, such as "The barking of the dog caused the cat to mew." Purely temporal relationships can generally be considered quantitative relations, as discussed above.

Two issues have proved particularly troublesome in temporal reasoning. The first is the "frame problem," first identified by McCarthy and Hayes (12). In order to reason about the effect of an event on the world, it is necessary to assert not only what effects it has on the world but also what effects it does not have. For instance, in reasoning about the event of the ball falling from the table to the ground, it is not only necessary to deduce that, at the end, the ball was on the ground but it is also necessary to deduce that the ground and the table were in the same place as before, that the ball was still the same color, that the robot observing this was still working, etc. In closed-world systems in which the state is characterized by a fixed number of independent facts, the problem can be handled by an axiom schema that an event changes only those facts that are specified. In open-world systems the problem is much harder. The problem is particularly acute in the situation calculus, but it persists in altered form in other ontologies as well.

The second problem is the "cross-world identification problem," which is common to many applications of possible-worlds semantics. It arises in two forms in temporal reasoning. The first problem involves identifying the same object in two different situations. If a car has all its parts replaced one by one, for example, at what point, if ever, does it cease to be the same car? The second, more difficult, problem involves identifying the same event across possible courses of events. If one possible course of history includes the event of Jim being run over by a truck while crossing the street, a second includes the event of his being run over by a car in the same act of his crossing the street, and a third has him dying in his bed 50 years later, which of these are the same event of Jim's death? These problems may seem artificial but actually form serious obstacles in stating causal laws.

**The Physical World.** Unlike the science of physics, which aims at a simple description of the underlying structure of physical reality, the commonsense theory of physics must try to describe and systematize physical phenomena as they appear and as they may most effectively be thought about for everyday purposes. The problems addressed in physical reasoning include predicting the future history of a physical system, explaining the behavior of a physical system, planning physical actions to carry out a task, and designing tools to serve a given purpose.

The representation used in physical reasoning depends strongly on the ontology used for time. If Hayes's histories are used, physical descriptions are descriptions of histories, and physical laws are laws constraining the interactions of histories. The representational problem then comes down to the

method used to describe individual histories and their spatio-temporal relations. If a situation-based system like McDermott's is used, there are four categories of representational issues: how are the static, unchanging aspects of a physical situation described; how is the time-varying state of a system described; how are events described; and how are causal laws described. For example, in electronic systems the structure is specified by the component characteristics and their connections; the state is specified by the voltages and currents in the system; and causal laws are specified in component equations and Kirchhoff's laws. In the Newtonian mechanics of rigid homogeneous objects the structure is specified by the shapes of the objects, the state is specified by the position and velocities of the objects, and the causal laws are the laws of mechanics.

Typical issues addressed in physical reasoning systems include the following:

**Closed Systems.** Probably the simplest kind of physical system to analyze is a closed system whose state is characterized by a set of one-dimensional parameters. The primary example of these is electronic networks; other examples are simple heat engines and hydraulic systems and systems with one-dimensional motion. The most common technique for analyzing such systems is to categorize the various states of the system into a finite collection of modes characterized by order relations on the parameters. The physical laws of the particular domain are used to calculate what sequences of modes are possible. The overall behavior of the system over time can then be expressed as a transition network showing the possible sequence of modes. Theories differ in the nature of the causal laws allowed, in particular, whether they are associated with physical components of the system or with processes acting on the system (33,34,49).

**Causal Explanations.** One function of a physical reasoning system is to explain to a human user how a physical phenomenon occurs. Such explanations are generally given in causal statements of the form "This causes that." (This is not the only function of causal statements of this kind.) Extracting such causal statements from a physical analysis may be difficult, particularly in systems like electronic systems, where effects propagate essentially instantaneously. De Kleer and Brown have devised a system of analysis that addresses this problem using a model of time with infinitesimal time intervals (34). Rieger and Grinberg use an extensive vocabulary of various kinds of causality found in physical mechanisms (50).

**Spatial Reasoning.** Spatial characteristics are critical in determining physical behavior. De Kleer (51) and Forbus (52) study the behavior of balls around obstacles in two dimensions. Novak (53) constructs a system for reasoning about rigid, two-dimensional objects with simple shapes. Hayes (48) presents a qualitative geometry for reasoning about liquids.

**Multiple Theories.** Physical knowledge can be viewed as consisting of many different theories that vary in domain (mechanics vs. electronics), accuracy (Newtonian mechanics vs. impetus theory), simplicity (thermodynamics vs. statistical mechanics), and viewpoint (energy and momentum vs. force). Given a problem it is sometimes necessary to choose among competing theories, to combine several theories, or to switch from one theory to another (53,54).

**Liquids.** Liquids present a tricky ontological problem since they are not individuated into separate objects. Hayes (48) proposes to individuate liquids by their containers and develops a physics of liquids in these terms. Similar problems about mass nouns generally are addressed by Bunt (55).



**Intelligent Agents.** To cope with a world containing intelligent creatures—itsself, even if no others—an intelligent system must have some understanding of their behavior and their mental life. The theory of intelligent agents is most important in story understanding, since stories about inanimate objects are rare and boring, and in planning, since plans and goals are necessarily associated with intelligence and many plans involve changing one's own mental state or that of someone else. A commonsense theory of mental states must deal with many different issues, including emotions, actions, plans, goals, and propositional attitudes.

The first of these categories, emotions, had not been extensively studied in the AI literature. The most careful study of emotions is probably that of the BORIS (qv) system (56). Here assertions about emotional states are predicates on seven arguments: the character feeling the emotion, the situation, whether the emotion is a positive one, the intensity of the emotion, the object of the emotion (optional), the goal situation giving rise to the emotion, and whether the situation was previously expected. Dyer (56) provides a number of plausible rules describing how success or failure of goals affect emotion and how emotions affect actions and goals (see Emotion modeling).

**Actions.** An action is an event that occurs at the volition of an intelligent agent. A primitive action is one that can be executed directly by the agent without further analysis. If the agent is a robot, primitive actions are generally taken to be movements of the effectors and uses of the sensors. Various primitives have been used to describe robot actions; they range from dealing directly with hardware motor operations to describing merely starting and ending positions of an effector (57). The analysis of sensor actions is less well developed.

Human beings, being more complex than robots, have a greater possible range of actions. The best known analysis of human actions is Schank's conceptual-dependency (qv) theory (11,58). This theory proposes 11 primitive acts: MOVE, to move a body part; INGEST, to consume an object; EXPEL, to emit an object; PROPEL, to exert force on an object; GRASP, to grasp an object; SPEAK, to make a noise; PTRANS, to move an object; ATTEND, to focus a sense organ; MBUILD, to think; MTRANS, to convey a thought; and ATRANS, to transfer ownership or control. Acts are related to each other and to world states via four types of causal links: an action may result in a state; a mental action can motivate a nonmental action; an event may initiate a mental action; or a state may enable an event. By combining these actions with causal and temporal links, it is possible to represent a significant class of simple stories.

**Plans and Goals.** The structure of plans and goals have been analyzed in great depth in studies both of planning and of story understanding. A fully articulated plan is made up of primitive actions connected by an unambiguous control structure. Such a plan is a program that can be interpreted directly by the robot as instructions to effectors and sensors without the need for further reasoning. A partially articulated plan has a more complex structure, since actions, tests, control structures, and groups of these may be only partially known. Moreover, a partially articulated plan may have associated with it an explanation of how it works. This explanation is used in further development of the plan. The nature of these explanations and incomplete specifications is a matter of current research (14,59). Also associated with a theory of plans must be a representation for the costs of executing a plan and

for the interactions between plans, favorable and unfavorable (60).

There are many theories of goals. In early work on problem solving a goal was a state to be achieved or a proposition to be made true (61,62). In real life there are many other kinds of goals. For purposes of story understanding, Schank and Abelson (11) divided goals into five classes: achievement goals, which are long-term ambitions; satisfaction goals, which are recurrent needs, such as food and sleep; entertainment goals; preservation goals, such as the need to preserve life, health, and property; and delta goals, which are state changes. These different categories vary in priority (e.g., preservation goals almost always take precedence over entertainment goals), in the structure of plans needed to achieve them, and probably in their underlying ontology. In more recent theoretical work on planning, goals have been replaced by the more general idea of tasks, which have not been given a clear ontological definition (63).

**Propositional Attitudes.** Many mental states are categorized as a binary relation, called an attitude, between the mind involved and some assertion about the world, called a proposition. For instance, John sees that it is raining, he hopes that the sun will come out, he fears that his cellar will flood, and he believes that his wife knows that the attic window is open. Here, "sees," "hopes," "fears," and "believes" are the attitudes, and "It is raining," "The sun will come out," "The cellar will flood," and "John's wife knows that the attic is open" are the propositions. As the last example shows, propositions may themselves be statements of propositional attitudes, to any depth of imbedding. Thus, a theory of propositional attitudes consists of answers to the two questions "What are the attitudes?" and "What, if anything, is a proposition?"

There is a large philosophical literature on the nature of propositions and propositional attitudes, and AI research in this area is largely grounded in antecedent philosophical theories. Two general approaches have been popular in AI. In the modal approach propositional attitudes are taken to be modal operators, and propositions are well-formed formulas, which may themselves contain modal operators. In the syntactic approach attitudes are taken to be first-order predicates, whose argument is a string (or Gödel number) that denotes a proposition in some language. The difference between these approaches is subtle but significant; it is roughly analogous to the difference between indirect and direct quotation ("He said that it was raining" vs. "He said, 'It is raining.'")

A possible-worlds semantics for the modal operator "A knows that *P*" was proposed by Hintikka, following Kripke's use of a possible-worlds semantics for necessity (64,65). In this theory each propositional attitude determines an "accessibility" relationship between possible worlds. Intuitively, a world *V* is accessible from world *U* with regard to agent *A*, if, as far as *A* knows in *U*, he might just as well be in *V*. More precisely, *A* knows proposition *P* in world *U* if and only if *P* is true in all worlds accessible from *U*. It can be shown (66) that many standard axioms of modal logic are strongly related to simple properties of the accessibility relationship. For example, the rule "If *A* knows *P* then *A* knows that *A* knows *P*" is implied by the statement that the accessibility relation is transitive. Possible-worlds semantics can be used for any of the propositional attitudes. A variant of possible-worlds semantics is the situation logic of Barwise and Perry (67) (not related to the situation calculus discussed above), in which partial descriptions of the world are used instead of possible worlds.

The major advantage of the syntactic approach is that it treats propositions as individuals. Since propositions are simply character strings, they may serve as logical constants, as the values of quantified variables, and as the arguments of functions and predicates. It is therefore possible to represent facts like "John believed what the salesman told him" or "Jim knew how Roosevelt was related to Churchill." In the modal approach the first fact cannot be represented at all, and the second cannot be represented unless the formal language provide a "relation-between" function that maps a pair of individuals onto relations like seventh-cousinhood. However, the syntactic approach has a serious problem in its capacity to express self-referential sentences. Using syntactic methods, it is easy to create sentences like "John believes that this sentence is false." Such sentences are clearly anomalous, and if a few plausible axioms about belief are asserted, they can be shown to lead to contradictions.

A major strength of the modal approach is the facility with which the possible-worlds semantics for knowledge can be combined with a possible-worlds semantics for time. This has been carried out by Moore (68), resulting in an elegant theory of knowledge and action. Moore also demonstrates that translating modal statements about knowledge into their possible-worlds equivalents makes it substantially easier to control inference.

One problem is troublesome for all formalizations of propositional attitudes: To what extent is it assumed that the believer sees the consequences of his own belief? Frequently, formal theories of belief adopt the rule "If  $A$  believes  $p$  and  $A$  believes  $p$  implies  $q$  then  $A$  believes  $q$ " as an axiom. Without this rule, theories of belief are generally too weak to support interesting inferences; moreover, the rule is unavoidably true if a possible-worlds semantics is adopted. However, it has some strange consequences: Everyone believes all mathematical theorems (since they follow from the null set); and anyone whose beliefs are not consistent believes everything. Various escapes from this problem have been suggested. The rule may be made merely a plausible, not a necessary, inference (68); belief may be divided into explicit belief (the statements the believer would actually assent to) and implicit belief (the statements that would follow from his explicit beliefs) (69); or limits may be placed on the kinds of inferences performed by a believer (69,70).

**Interpersonal Interactions.** The final domain of commonsense knowledge is the interactions between people and groups of people. This area is so diverse and so unstructured that it is impossible, in this entry, to do more than indicate some of the areas that have been examined:

**Language and Communication.** A number of researchers (71-73) have studied the logic of communicating facts and making requests between friendly agents. A communication is modeled as a belief of the first agent that he causes to be believed by the second; a request is a goal of the first agent that he causes to be adopted by the second.

**Stereotyped Social Situations.** Social situations that are governed by conventional rules, such as restaurant meals or job interviews, have been represented with a fair degree of success by frames, called scripts (qv), specifying a temporal sequence of events (11). The exact content of scripts, the interactions between them, and their relationships to deeper theories of human behavior based on affects, plans, and goals are current areas of research (74,75).

**Personal Relationships.** A major determinant of the appropriate or expected behavior of one person toward another is their personal relationship. When the people are engaged in a stereotyped situation, such as described above, the relationship is largely defined by the relevant script. Otherwise, the relationship is a complex of many different components: their emotions toward each other, the interaction of their goals, conventional rules governing their relative social positions, relative degree of authority, etc. (11,56,76).

**Possession.** The basic theory of possession is simple; the reality is extremely complex. Ninety-nine percent of the time, possession is adequately categorized as a relation between a person and a physical object obeying simple rules such as "Only one person owns an object at a time," "Possession can be changed from one person to another at any time if both parties are willing," "The possessor of an object has sole use of the object," etc. The actual theory of possession—in which possession is essentially determined by society, the object of possession need not be physical, the possessor need not be a single person or even a set of people, etc.—is perhaps beyond the scope of common sense and is certainly dependent on a detailed theory of law. It may be noted that early versions of conceptual-dependency theory distinguished several different types of possession until it was found that the distinction had no important consequences for commonsense inferences (77).

## BIBLIOGRAPHY

1. M. Minsky, A Framework for Representing Knowledge, in R. Brachman and H. Levesque (eds.), *Readings in Artificial Intelligence*, Kaufmann, Los Altos, CA, pp. 245-262, 1985.
2. P. Hayes, In Defense of Logic, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, pp. 559-565, 1977.
3. P. Hayes, The Naive Physics Manifesto, in D. Michie (ed.), *Expert Systems in the Micro-electronic Age* Edinburgh University Press, Edinburgh, Scotland, 1978. Reprinted and revised in J. Hobbs and R. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, pp. 1-36, 1985.
4. D. McDermott, "Tarskian semantics, or no notation without denotation!" *Cog. Sci.* 2(3), 277-282 (1978).
5. A. Newell, "The knowledge level," *AI Mag.* 2(2), 1-20 (1981).
6. Y. Wilks, Philosophy of Language, in E. Charniak and Y. Wilks (eds.), *Computational Semantics*, North-Holland, Amsterdam, 1976.
7. J. Hobbs, Introduction to J. Hobbs and R. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, 1985.
8. D. Israel, A Short Companion to the Naive Physics Manifesto, in J. Hobbs and R. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, pp. 427-448, 1985.
9. R. Moore, The Role of Logic in Knowledge Representations and Commonsense Reasoning, in R. Brachman and H. Levesque (eds.), *Readings in Knowledge Representation*, Kaufmann, Palo Alto, CA, pp. 335-342, 1985.
10. R. Montague, *Formal Philosophy*, Yale University Press, New Haven, CT, 1974.
11. R. Schank and R. Abelson, *Scripts, Plans, Goals, and Understanding*, Erlbaum, Hillsdale, NJ, 1977.
12. J. McCarthy and P. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 4, Edinburgh University Press, Edinburgh, pp. 463-502, 1969.
13. L. Wos, R. Overbeek, E. Lusk, and J. Boyle, *Automated Reason-*

- ing: *Introduction and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
14. N. Nilsson, *Principles of Artificial Intelligence*, Kaufmann, Palo Alto, CA, 1980.
  15. L. Schubert, "Extending the expressive power of semantic networks," *Artif. Intell.* 11, 45–83 (1978).
  16. G. Hendrix, Expanding the Utility of Semantic Networks through Partitioning, *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, pp. 115–121, 1975.
  17. R. Brachman, On the Epistemological Status of Semantic Networks, in R. Brachman and H. Levesque (eds.), *Readings in Knowledge Representation*, Kaufmann, Palo Alto, CA, pp. 191–216, 1985.
  18. T. Winograd (ed.), *Artif. Intell.* 13(1,2) (1980). Special volume on nonmonotonic logic.
  19. D. McDermott, "Nonmonotonic logic II: Nonmonotonic modal theories", *J. Assoc. Comput. Mach.* 29(1), 33–57 (1982).
  20. R. Moore, "Semantical considerations on nonmonotonic logic," *Artif. Intell.* 25, 75–94 (1985).
  21. J. de Kleer, J. Doyle, G. Steele, and G. Sussman, Explicit Control of Reasoning, Technical Memo 427, MIT AI Lab, Cambridge, MA, 1977.
  22. J. Doyle, A Truth-Maintenance System, Technical Memo 521, MIT AI Lab, Cambridge, MA, 1979.
  23. D. McAllester, An Outlook on Truth Maintenance, Technical Memo 551, MIT AI Lab, Cambridge, MA, 1980.
  24. P. Cheeseman, In Defense of Probability, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 1002–1109, 1985.
  25. E. Charniak, The Bayesian Basis of Common Sense Medical Diagnosis, *Proceedings of the Third National Conference on Artificial Intelligence*, Washington, DC, pp. 70–73, 1983.
  26. L. Zadeh, "Fuzzy algorithms," *Inform. Ctrl.* 12, 92–102 (1963).
  27. G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ, 1976.
  28. P. Hayes, The Logic of Frames, in R. Brachman and H. Levesque (eds.), *Readings in Knowledge Representation*, Kaufmann, Palo Alto, CA, pp. 287–296, 1985.
  29. E. Charniak, "A common representation for problem-solving and language-comprehension information," *Artif. Intell.* 16(3), 225–255 (1981).
  30. D. Bobrow and T. Winograd, "An overview of KRL, a knowledge representation language," *Cog. Sci.* 1(1), 3–46 (1977).
  31. R. Roberts and I. P. Goldstein, The FRL Manual, MIT AI Lab Technical Report 409, Cambridge, MA, 1977.
  32. J. Allen, "Maintaining knowledge about temporal intervals," *CACM*, 26, 832–843 (1983).
  33. K. Forbus, Qualitative Process Theory, in D. Bobrow (ed.), *Qualitative Reasoning about Physical Systems*, MIT Press, Cambridge, MA, pp. 85–168, 1985.
  34. J. de Kleer and J. S. Brown, A Qualitative Physics Based on Confluences, in D. Bobrow (ed.), *Qualitative Reasoning about Physical Systems*, MIT Press, Cambridge, MA, pp. 7–84, 1985.
  35. J. de Kleer and D. Bobrow, Qualitative Reasoning with Higher-Order Derivatives, *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, pp. 86–91, 1984.
  36. B. Kuipers, The Limits of Qualitative Simulation, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 128–136, 1985.
  37. T. Dean, Temporal Reasoning Involving Counterfactuals and Disjunctions, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 1060–1062, 1985.
  38. D. McDermott and E. Davis, "Planning routes through uncertain territory," *Artif. Intell.* 22, 107–156 (1984).
  39. E. Davis, Representing and Acquiring Geographic Knowledge, Technical Report 292, Computer Science Department, Yale University, New Haven, CT, 1984.
  40. B. Kuipers, "Modelling Spatial Knowledge," *Cog. Sci.* 2(2), 129–154 (1978).
  41. R. Brooks, "Symbolic reasoning among 3-D models and 2-D images," *Artif. Intell.* 17(1–3), 285–348 (1981).
  42. D. Marr, *Vision*, W. H. Freeman, San Francisco, 1982.
  43. D. Ballard and C. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
  44. A. A. G. Requicha, "Representations for rigid solids: Theory, methods, and systems," *ACM Comput. Serv.* 12(4), 437–464 (1980).
  45. J. McCarthy, Programs with Common Sense, in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, pp. 403–418, 1968.
  46. D. McDermott, "A temporal logic for reasoning about processes and plans," *Cog. Sci.* 6, 101–155 (1982).
  47. J. Allen, "Towards a general theory of action and time," *Artif. Intell.* 23, 123–154 (1984).
  48. P. Hayes, Naive Physics 1: Ontology for Liquids, in J. Hobbs and R. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, pp. 71–108, 1985.
  49. B. Williams, Qualitative Analysis of MOS Circuits, in D. Bobrow (ed.), *Qualitative Reasoning about Physical Systems*, MIT Press, Cambridge, MA, pp. 281–346, 1985.
  50. C. Rieger and M. Grinberg, The Declarative Representation and Procedural Simulation of Causality in Physical Mechanisms, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, pp. 250–256, 1977.
  51. J. de Kleer, Qualitative and Quantitative Knowledge in Classical Mechanics, Technical Report 352, MIT AI Lab, Cambridge, MA, 1975.
  52. K. Forbus, A Study of Qualitative and Geometric Reasoning in Reasoning about Motion, Technical Report 615, MIT AI Lab, Cambridge, MA, 1979.
  53. G. Novak, Representation of Knowledge in a Program for Solving Physics Problems, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA pp. 286–291, 1977.
  54. J. Hobbs, Granularity, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 432–435, 1985.
  55. H. Bunt, The Formal Representation of (Quasi-)Continuous Concepts, in J. Hobbs and R. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, pp. 37–70, 1985.
  56. M. Dyer, *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension*, MIT Press, Cambridge, MA, 1983.
  57. T. Lozano-Perez, Robot Programming, Technical Report 698, MIT AI Lab, Cambridge, MA, 1982.
  58. R. Schank, *Conceptual Information Processing*, North-Holland, Amsterdam, 1975.
  59. E. Charniak and D. McDermott, *Introduction to Artificial Intelligence*, Addison-Wesley, Reading, MA, 1985.
  60. R. Wilensky, *Planning and Understanding*, Addison-Wesley, Reading, MA, 1983.
  61. A. Newell and H. Simon, GPS, A Program that Simulates Human Thought, in E. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, pp. 279–298, 1963.
  62. E. Sacerdoti, *A Structure for Plans and Behavior*, Elsevier, New York, 1977.

63. D. McDermott, Reasoning about Plans, in J. Hobbs and R. Moore (eds.), *Formal Theories of one Commonsense World*, Ablex, Norwood, NJ, pp. 269–318, 1985
64. J. Hintikka, *Knowledge and Belief*, Cornell University Press, Ithaca, NY, 1962.
65. S. Kripke, "Semantical analysis of modal logic," *Zeitschr. Math. Log. Grundle. Math.* **9**, 67–96 (1963).
66. S. Kripke, Semantical Considerations on Modal Logic, in L. Linsky (ed.), *Reference and Modality*, Oxford University Press, London, pp. 63–72, 1971.
67. J. Barwise and J. Perry, *Situations and Attitudes*, MIT Press, Cambridge, MA, 1983.
68. R. Moore, A Formal Theory of Knowledge and Action, in J. Hobbs and R. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, pp. 319–358, 1985.
69. H. Levesque, A Logic of Explicit and Implicit Belief, *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, pp. 198–202, 1984.
70. K. Konolidge, Belief and Incompleteness, in J. Hobbs and R. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, pp. 359–404, 1985.
71. C. R. Perrault and J. Allen, "A plan-based analysis of indirect speech acts," *Am. J. Computat. Ling.* **6**(3–4), 167–182 (1980).
72. D. Appelt, Planning Natural-Language Utterances to Satisfy Multiple Goals, SRI Artificial Intelligence Center Technical Note 259, Menlo Park, CA, 1982.
73. W. Lehnert, *The Process of Question Answering*, Erlbaum, Hillsdale, NJ, 1978.
74. R. Schank, *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*, Cambridge University Press, Cambridge, U.K., 1982.
75. J. Kolodner, Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model, Research Report 187, Yale University, Department of Computer Science, New Haven, CT, 1980.
76. W. Lehnert, Affect and Memory Representations, *Proceedings of the Third Conference of the Cognitive Science Society*, Berkeley, CA, pp. 78–83, 1981.
77. R. Schank, Yale University, personal communication, 1981.

### General References

- D. Bobrow (ed.), *Qualitative Reasoning about Physical Systems*, MIT Press, Cambridge, MA, 1985. Current papers on physical reasoning.
- R. Brachman and H. Levesque (eds.), *Readings in Knowledge Representation*, Kaufmann, Palo Alto, CA, 1985. Reprints of many classic articles. Extensive bibliography.
- E. Charniak and D. McDermott, *Introduction to Artificial Intelligence*, Addison-Wesley, Reading, MA, 1985. Chapters 1, 6, and 7 give an excellent introduction to knowledge representation and commonsense reasoning.
- J. Hobbs and R. Moore, *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, 1985. Collection of state-of-the-art papers.

E. DAVIS  
New York University

### REASONING, DEFAULT

AI systems use reasoning mechanisms to derive conclusions on the basis of facts and rules stored explicitly in their databases. The desired conclusions may bear one of two very different

relationships to the information from which they are obtained. In the first case the conclusions follow logically from the information present in the system. That is, the system contains assertions of facts or relations among facts, and if these assertions are all true, the desired conclusion is guaranteed also to be true. When this relationship between the system's beliefs and its desired conclusions holds, those conclusions can be deduced by reasoning techniques based on standard logics, most frequently first-order predicate logic (qv).

But when AI systems are applied to situations more closely resembling those in which people function, it becomes desirable for them to draw conclusions that are less tightly based on the information at hand. In these cases the system may have information that is relevant to a conclusion and that makes it reasonable to presume that the conclusion is true but does not logically entail it. People routinely act on the basis of such partial information, and in many situations an AI system that cannot act on such information becomes paralyzed. Hence it becomes useful for systems to use inference mechanisms that reach conclusions not accessible using standard logic. The term "default reasoning" has come to be used for patterns of inference that permit drawing conclusions suggested but not entailed by their premises. Different researchers have applied some kind of default reasoning to a variety of situations, each with its own nature or requirements and each involving different kinds of generalizations with their own internal structures and rules. It follows that proposed default-reasoning systems also tend to vary, depending on the situations and generalizations on which their designers focus. Default reasoning plays a particularly important role in two different kinds of situations which it may be helpful to distinguish: situations in which systems must reason from incomplete information and those in which systems must reason using uncertain rules (see Reasoning, plausible).

Reasoning from incomplete information most often takes the form of using some kind of nonuniversal generalizations to bridge specific gaps in knowledge. These generalizations often represent typicality judgments: to use the example that has become standard in these discussions, when people say that birds fly, they mean that typical birds do. This is not the same as saying that most birds fly or that there is a probability over some threshold that something can fly given that it is a bird. First, such probabilistic interpretations of generalizations are frequently false; consider the situation in the late spring, for instance, when infants outnumber adults in the bird population. Second, statistical analyses work on a different basis from more familiar reasoning. Briefly, statistical analysis begins by establishing a space of outcomes, which are mutually exclusive observations or results in an experimental setting. That is, given any experiment, one and only one outcome will be observed. Beginning with this outcome space, the methods of statistics evaluate (among other things) the probabilities that the results of a given experiment will fall into one or another subset of the outcome space; these sets of outcomes are called events. More familiar reasoning styles (including predicate logic) begin with a "space" of atomic propositions, which by definition are mutually permissive (i.e., the truth of one atomic proposition entails nothing about the truth or falsehood of any other). Given this difference between a mutually exclusive outcome space and a mutually permissive propositional base, it should not be surprising that the rules for propagating probabilities are incompatible with the predicate-logic-style inferences that are generally viewed as underlying

reasoning from generalizations. In other words, typicality-based reasoning is not just informal probability reasoning. (For an analysis of typicality as category cue validity and the differences between that statistical notion and probability, see Refs. 1–3.)

In some situations, the problem is less one of missing information or of making presumptions based on notions of typicality than one of dealing with rules that are known to be uncertain in their application. Such situations include genuinely probabilistic cases, where event probabilities are being measured on the basis of information about base probabilities, joint probabilities, and the like. In these cases information may be complete. The statement "A fair toss of a coin results in an outcome of heads with a probability of 0.5" does not represent missing knowledge concerning the behavior of tossed coins. On the contrary, it represents the most complete statement possible of that behavior. The inference from coin tosses to outcomes is inherently uncertain, but follows rules that allow making very precise judgments concerning the probability of any particular event (say, a string of 50 tosses of which precisely half are heads). Contexts of incomplete information frequently also involve uncertainty and vice versa. But what is going on in the judgments is not quite the same, and so modeling the two may require more than one specialized inference mechanism.

Two other kinds of situation frequently arise in connection with default reasoning and may be important to distinguish from the foregoing cases. The first, and for many researchers the most interesting, involves what might be called belief selection. Suppose that a system is faced with several alternative propositions, any one of which can be added to the system's current beliefs without entailing a contradiction but which cannot all be added at once without entailing a contradiction. The system could choose to believe none of the alternatives (i.e., to remain agnostic) or it could select one to believe. If the current state of information in the system clearly prefers one of the options on the basis of suggestive partial information or on the basis of some uncertain rule or rules, this situation collapses into one of the previous two. But that may not be the case. Then how does the system choose? This question encompasses several very deep issues, since the problem here includes not only deciding what the current state of knowledge and belief suggests or entails but also determining *a priori* what it is rational to believe. This latter question has both deep philosophical implications and substantial practical complications. There may be some propositions it is not rational to believe even in the presence of evidence for them and others it is rational to believe even in the absence of evidence. Furthermore, the extent to which agnosticism is a rational choice remains unresolved.

The second, related problem is belief revision (qv). When new information that conflicts with old beliefs enters a system, how does the system decide which of its old beliefs to change and how to change them? This problem is particularly difficult when some of the system's old beliefs were arrived at through default reasoning, since in those cases clearly the system would rather reject previous default conclusions than reject better grounded information. Once again, rational choice of commitments becomes a focal concern.

Although belief selection and belief revision represent important research areas, it is not clear that either falls under the heading of default reasoning. It is important to distinguish between selecting premises, drawing inferences on the basis of

premises, and deciding how to alter premise sets when belief systems are found inconsistent. The second task is clearly a reasoning task, and when the inferences drawn are suggested rather than entailed, it is a default-reasoning task. But what about the first? By definition, reasoning is the process of drawing conclusions from premises. Premise selection is thus a prerequisite for the reasoning process, not its product. Although there may be grounds of various kinds (some of them rational, others probably not) for selecting some beliefs rather than others, the act of commitment to premises itself is not one of reasoning, default or otherwise. The third task, that of restoring consistency in a belief structure that has become inconsistent, has four parts: (a) identifying which old beliefs are premises, which are deductive conclusions from the premises, and which are presumptions suggested by the premises; (b) identifying logical requirements for restoring consistency; (c) deciding which old premises (if any) to change and how; and (d) updating deductive conclusions and default presumptions to reflect those changes. Of these four, (a) is essentially a book-keeping requirement; it reflects information that ought to be represented in any system intended to accomplish this kind of task. Parts (b) and (d) are ordinary reasoning problems (though (d) may involve default reasoning if the original system did), and (c)—the hard part—is premise selection in a new guise. For these reasons, approaches to these two problems (such as dependency-directed backtracking (qv)) are not discussed here (for a discussion of these approaches, see Belief revision).

What, then, must a default-reasoning system provide? At least three things stand out as necessary. First, the system's status must be clearly presented. There are at least three options here. A default-reasoning system may be presented as a system of logical inference, i.e., as a mechanism for inferring what propositions follow from what other propositions by virtue of their forms (see Inference). Alternatively, the reasoning system may embody some form of probabilistic model in which probabilities of events are calculated from base probabilities over an outcome space (see Bayesian decision methods). Finally, the system may embody some other form of reasoning capacity different from both of the above and making use of other kinds of information than either probability measures or formal properties of propositions. Understanding a default-reasoning system involves knowing which of these three options it represents. In addition, the system must provide inference mechanisms for deriving new consequences. To many researchers, this is the heart of any default-reasoning system. The inference mechanisms must have at least the following two properties to be useful in AI systems: they must be implementable at least in principle, and they must involve some analog of the usual soundness property for logics, i.e., there must be some way of showing that the conclusions they permit systems to reach are in some reasonable sense warranted by the information from which they are derived. Third, there must be some form of semantics provided for interpreting the meaning of the inferences made. This can be viewed as a prerequisite for showing that the inference mechanism has the second property mentioned above: the semantics for the default-reasoning system provides the definitions of meaning and consequence relative to which conclusions of inferences must be warranted consequences of their premises.

This entry reviews several of the approaches to default reasoning that have been proposed. These fall into two general categories: logic-based approaches and measure-based ap-

proaches. Logic-based approaches have developed from several standpoints. Non-monotonic logics attempt to establish general form-based rules for inferences with the kinds of properties that default reasoning appears to require. Other proposals include nonmonotonic reasoning mechanisms for extending normal logical inferences and monotonic approaches to a logic for default reasoning. Approaches based on numerical measures divide into two groups. Probabilistic approaches deal with default reasoning either as an instance of Bayesian analysis or as some other probability-based mechanism. Non-probabilistic measure-based approaches attempt to retain the numerical nature of probability reasoning but divorce inference from the outcome space basis which statistics presupposes and wed it instead to a propositionally based and propagated mechanism.

### Logic-Based Approaches

**Nonmonotonic Logic.** In all standard (and most nonstandard) systems of logic, if a proposition  $C$  can be derived from a set of propositions  $S$ , and if  $S$  is a subset of  $S'$ , then  $C$  can also be derived from  $S'$ . That is, adding information without eliminating any cannot eliminate deductive entailments: as a system's premises increase, its possible conclusions at least remain constant (and more likely increase). Deductive systems with this property are called monotonic. The dominant approach to default reasoning to date involves viewing reasoning with uncertain or incomplete information as a nonmonotonic process: inferences that are reached on bases of these kinds may need to be discarded in the light of new evidence. Hence such inferences are freer than the more familiar logical inferences in that the evidence on which they are based does not constrain their truth. Nevertheless, they follow rules based on the forms of the propositions involved. Viewing these as rules of inference, justifying drawing conclusions, leads to attempts to develop nonmonotonic logics to model default reasoning.

One of the first such logics (see Reasoning nonmonotonic) was presented in 1980 by McDermott and Doyle (4) and was further refined by McDermott in 1982 (5). This logic extends ordinary first-order logic by introducing a modal operator  $\mathbf{M}$  whose intuitive meaning is deductive consistency. That is,  $\mathbf{M}p$  is intended to mean that  $p$  is deductively consistent with all current beliefs in the sense that  $\sim p$  cannot be proved from the current axioms and assumptions. Clearly, then, the meaning of  $\mathbf{M}p$  will change with the state of current belief: the more that is known, the more that is ruled out. The particular propositions that the kinds of systems McDermott and Doyle had in mind focus on most are propositions of the general form  $(q \wedge \mathbf{M}p) \supset p$ , which are intended to capture the idea that if you know  $q$ , and you know no reason why not  $p$ , then you may reasonably infer  $p$ . To use an example from Ref. 4, from the theory given by the statements

1.  $(\text{it-is-noon} \wedge \mathbf{M}[\text{sun-is-shining}]) \supset \text{sun-is-shining}$ ,
2.  $\text{it-is-noon}$
3.  $\text{eclipse-is-happening} \supset \sim \text{sun-is-shining}$ ,

it is possible to infer sun-is-shining, but if the further axiom

4.  $\text{eclipse-is-happening}$

is added to this theory, sun-is-shining can no longer be inferred. Hence adding this feature to the inference system

makes it non-monotonic because previously permitted inferences are ruled out by the addition of knowledge. McDermott and Doyle present their system as a logic, refer to its inferences as proofs, and present a formal semantics to justify them, but it is important to understand that they do not view these proofs as establishing their conclusions' truth, even conditional on the truth of their premises. In their own words,

*the purpose of non-monotonic inference rules is not to add certain knowledge where there is none, but rather to guide the selection of tentatively held beliefs in the hope that fruitful investigations and good guesses will result. This means that one should not a priori expect nonmonotonic rules to derive valid conclusions independent of the monotonic rules. Rather one should expect to be led to a set of beliefs which while perhaps eventually shown incorrect will meanwhile coherently guide investigations" (6).*

Their primary concern throughout is not with valid proof but with reasonable procedures for truth maintenance, which they see as involving two aspects: preventing a database ever containing both  $\mathbf{M}p$  and  $\sim p$  for any  $p$ , and detecting and repairing inconsistencies by adding appropriate axioms. Since they are less concerned with preventing false hypotheses than with correcting impossibilities, their system takes a permissive stand, concentrating on ruling out assumptions that cause contradictions but permitting all others.

Reiter has developed an alternative logic for default reasoning (7-10) that generalizes his earlier closed-world assumption (11): For any "ground fact" consisting of a relation  $R$  and individuals  $x_1, \dots, x_n$ , if the system cannot prove  $R(x_1, \dots, x_n)$ , it may conclude  $\sim R(x_1, \dots, x_n)$ . The intuition behind this assumption is that many databases include all positive base facts in their domain: if a database representing airline flight information has no information on a certain flight, it is justified in concluding that no such flight exists. This concept is then extended to cover reasoning from generalities by treating exceptions as positive ground facts which, if absent, may reasonably be assumed false. One of the earliest uses of defaults in reasoning systems arises in frame-based systems (12,13) with the use of default values to provide attributes in unexceptional cases. On the basis of this kind of reasoning, Reiter interprets generalizations such as "Most birds fly" as follows:

*We take it to mean something like "If  $x$  is a bird, then in the absence of any information to the contrary, infer that  $x$  can fly." The problem then is to interpret the phrase "in the absence of any information to the contrary." The interpretation we adopt is "It is consistent to assume that  $x$  can fly." Thus "If  $x$  is a bird and it is consistent to assume that  $x$  can fly, then infer that  $x$  can fly" (14).*

Like McDermott and Doyle, Reiter's initial presentation used  $\mathbf{M}$  to represent this consistency. Thus, in his system, the rule given above is represented symbolically as follows:

$$\frac{\text{BIRD}(x): \mathbf{M}\text{FLY}(x)}{\text{FLY}(x)}$$

In later formulations the  $\mathbf{M}$  was dropped, so that "in general birds fly" would then be written as " $\text{BIRD}(x): \text{FLY}(x)$ ." The ":" functions as a (unary or binary) connective that permits the use of a specialized rule of inference to extend the underlying first-order logic. As with McDermott and Doyle, consistency is



taken to mean deductive consistency with both the first-order facts about the world and all other beliefs sanctioned by all other default rules in force. Reiter does appear to distinguish results of default reasoning from other assertions:

*Notice that if FLY(tweety) is inferred by default then the assertion FLY(tweety) has the status of a belief; it is subject to change, say by the subsequent discovery that tweety is a penguin. We can then reinterpret the default rule . . . as saying that "If x is a bird and it is consistent to believe that x can fly then one may believe that x can fly" (15).*

If the system as a whole distinguishes the representations of "one may believe that x can fly" and "x can fly," however, that distinction seems not to be reflected in the workings of the logic Reiter proposes. (For the importance of this distinction, see Refs. 16 and 17.) Reiter's interpretation of defaults leads to nonmonotonicity exactly as McDermott and Doyle's does, although the focus of interest, and hence the stress in the examples, is different. The primary difference between Reiter's logic and McDermott's lies in the different status given to non-monotonic rules (propositions involving the operator *M*). McDermott views these as modal propositions within the language of a first-order logic, whereas Reiter takes a different approach:

*We avoid modal logic or any modal concepts entirely and work only within a first order framework. For us defaults function as meta-rules whose role it is to further complete an underlying incomplete first order theory whereas for McDermott and Doyle defaults are formulae of a modal logic (18).*

Reiter views defaults as "rules for extending an underlying incomplete first order theory" (19), and focuses on providing a formal definition of the extensions to that theory that result from adding defaults. He then provides a proof theory for default logic that, given a default theory and a particular proposition, determines whether that proposition belongs to one of the extensions of the default theory. For a further discussion of these and other formalisms of nonmonotonic logic, including a discussion of the semantics for such systems, see Reasoning, non-monotonic.

**Circumscription.** John McCarthy's circumscription (qv) mechanism (20–22) was introduced in 1980 to provide a means for representing and reasoning from commonsense knowledge in a suitable formalism based on ordinary mathematical logic. According to McCarthy, the basic problem was one of what he calls qualification:

*It seemed that in order to fully represent the conditions for the successful performance of an action, an impractical and implausible number of qualifications would have to be included in the sentences expressing them. For example, the successful use of a boat to cross a river requires, if the boat is a rowboat, that the oars and rowlocks be present and unbroken, and that they fit each other. Many other qualifications can be added, making the rules for using a rowboat almost impossible to apply, and yet anyone will still be able to think of additional requirements not yet stated (23).*

In order to deal with this problem, McCarthy introduces the concept of circumscribing predicates. The circumscription of a

predicate is a proposition which represents the assumption that "the objects that can be shown to have a certain property *P* by reasoning from certain facts *A* are all the objects that satisfy *P*" (23; emphasis in original). That is, circumscription provides a formalism for drawing conclusions based on the assumption that all the information that would be needed to derive exceptions is present.

Although Reiter and McDermott's defaults might be called permissive in that they allow systems to conclude anything that is suggested by some default rule and consistent with the current knowledge of the system, McCarthy's rules take a restrictive approach. Instead of looking at all consistent extensions of the current state of the knowledge, circumscription restricts itself to minimal extensions. An extension of a theory is minimal if it represents a model that satisfies the theory and has no substructure that does. Thus a minimal extension is one that contains exactly as many things as needed to make sense of the knowledge in the theory. The basic intuition here derives from Occam's Razor: only those things exist whose existence is required by our understanding of the situation. McCarthy treats exception conditions as things; this allows him to draw default-style conclusions, since the minimal extensions of knowledge at any time will contain no things, and hence no exception conditions, that one's knowledge does not entail.

The central concept of McCarthy's formalism is the circumscription of a predicate (or set of predicates) within a sentence. The purpose of the circumscription is to give a context relative to which circumstances can be considered unexceptional for making default conjectures: "Conclusions derived from circumscription are conjectures that *A* includes all the relevant facts and that the objects whose existence follows from *A* are all the relevant objects" (24). By definition, the circumscription of a predicate *P* in a sentence *A(P)* that contains that predicate is given by the following sentence scheme, where  $\Phi$  is a predicate parameter:

$$[A(\Phi) \wedge \forall \bar{x}(\Phi(\bar{x}) \supset P(\bar{x}))] \supset \forall \bar{x}(P(\bar{x}) \supset \Phi(\bar{x}))$$

To use McCarthy's blocks-world example, the circumscription of "isblock" in the sentence "isblock *A*  $\wedge$  isblock *B*  $\wedge$  isblock *C*" gives a scheme that says in essence that given any property  $\Phi$ , if the objects *A*, *B*, and *C* have  $\Phi$ , and if anything that has  $\Phi$  is a block, then everything that is a block has  $\Phi$ . In particular, consider the property of being (identical to) either *A*, *B*, or *C*. Clearly each of *A*, *B*, and *C* have the property, and clearly anything that has the property is a block (since *A*, *B*, and *C* are all blocks). So the circumscription scheme lets one conclude, as a conjecture, that every block has the property of being either *A*, *B*, or *C*. Conjectures justified by circumscription may become unjustified if new information is added to the system: if the system learns that *D* is a block, and if that clause is conjoined to the sentence within which "isblock" is circumscribed, it can no longer conclude that anything that is a block is either *A*, *B*, or *C*.

McCarthy distinguishes the reasoning mechanism circumscription provides from logic: "Circumscription is not a 'non-monotonic logic.' It is a form of nonmonotonic reasoning augmenting ordinary first order logic" (25). The reason for this distinction is that the familiar systems of logic, which McCarthy believes should be adhered to, provide rules for inferring conclusions from premises that logically entail them, whereas circumscription produces conjectures, which do not follow by logic. The approach embodied by circumscription can

nonetheless reasonably be described as logic-based for several reasons. First, McCarthy's aim is precisely to be able to express commonsense knowledge and provide for commonsense reasoning (qv) in a setting derived from a close variant of standard first-order predicate logic. To this end, his circumscription formalism defines the circumscription of a proposition as a first-order schema that is to be construed by means of and used in conjunction with the normal rules of first-order predicate logic. Second, the circumscription formalism can be viewed as a means for making explicit a particular kind of underlying assumption about knowledge, adding it into the system's premises, and then performing ordinary logical inferences, with the proviso that since the underlying assumption in question is less a belief than a heuristic for making guesses and since adding information will rule out many of those guesses, the conclusions that can be reached will change—in particular, be restricted—as knowledge is gained.

**Monotonic Approaches.** Nonmonotonic logics have received criticism on both mathematical (see especially Ref. 26 for mathematical criticisms of the work of McDermott and Doyle) and nonmathematical (see especially Ref. 27; also Refs. 16 and 28) grounds. Recently, several researchers have begun to introduce monotonic reasoning schemes that may be applicable to default reasoning. Cohen has published preliminary work on a default-reasoning system based on the concept of endorsements (29–31). This proposal concentrates on the problem of distinguishing options, and argues for extending ordinary logical reasoning by considering multiple sources of support (or counter-support) rather than any single measure of possibility, evidence, or credibility. Work on endorsements is still in the early stages, however, and although the basic principles behind this approach have been developed, as yet no precise inference mechanism has been put forward. Levesque has also made substantial contributions, especially in his dissertation [(32); see also Refs. 33 and 34]. Nutter has presented a conservative extension of standard predicate logic (35) that explicitly distinguishes presumptions (used in expressing default generalizations and their conclusions) from assertions; i.e., within this logic the default conclusion "Presumably Tweety flies" is a separate proposition from, and not inconsistent with, the assertion "Tweety flies." Under this view, conclusions from default reasoning are always guarded propositions, which continue to be true when new information is added that is incompatible with their unguarded versions. Hence from "Generally birds fly," "Tweety is a bird," "No ostriches fly," and "Tweety is an ostrich," it follows that "There is reason to believe that Tweety flies, but in fact he doesn't," a claim that is both true and in some contexts useful. (It can be used, e.g., to detect features that are atypical and therefore interesting; see Ref. 17.) Since guarded conclusions are not lost when new premises are added, the resulting logic does not satisfy the definition of nonmonotonic logics.

### Measure-Based Approaches

A measure-based approach to default reasoning is one that applies some form of numerical weights (most frequently real numbers in the interval from 0 to 1) to the information in the database, propagates those measures through a reasoning process, and reaches a conclusion on the basis of that propagation. Most frequently, the weights are viewed as measuring something related to a probability (though not always in the statis-

tical sense). Propositions with a weight of 1 are typically taken as certain and propositions with a weight of 0 as certainly false; weights between 0 and 1 represent degrees of probability, uncertainty, vagueness, or some similar concept. The conclusion of such a chain of reasoning is typically a proposition found to have an "acceptable" weight; depending on the approach, this may mean the greatest weight of some predefined set of alternatives, a weight near 1, a maximum of a set of alternatives that also meets some threshold, or some other similar criterion. This conclusion is frequently added to the system's knowledge base together with its derived weight. Since both the conclusion and the weight assigned to it depend on the current state of the system's information, both may change in the light of further information.

**Bayesian Analysis.** Although there are disputes in the literature about whether it is appropriate to model all cases of incomplete and uncertain knowledge using probabilities or related concepts, it is clear that probability models are appropriate for some situations involving uncertainty. One AI system that uses Bayesian analysis (see Bayesian decision methods) is PROSPECTOR (36). It should be noted, however, that implementing probabilistic analysis requires care and attention to the principles as well as the formulas of the field of statistics. For instance, as remarked above, statistical analysis does not start with propositions; it starts with outcomes. The difference between the kinds of propositions on which logical rules of inference are based and those that express outcomes is dramatic: the atomic sentences of a first-order logic are by definition mutually permissive (any two may simultaneously be true) and outcomes (but not events) are by definition mutually exclusive (no two propositions expressing outcomes can simultaneously hold). The formulas for determining event probability are based on combinations of outcome probabilities; there is no reason to expect them to hold if they are applied instead to combinations of probabilities for arbitrary propositions.

However, although some AI systems can clearly benefit by carefully conducted probability analyses, there are sound reasons to believe that default reasoning in general cannot be replaced by probability reasoning. At present, probably the most serious objection to using Bayesian analysis arises from its requiring prior probabilities for all its hypotheses and assertions: where will the numbers come from? This objection was raised in print by McCarthy and Hayes in 1969:

*On numerous occasions it has been suggested that the formalism take uncertainty into account by attaching probabilities to its sentences. We agree that the formalism will eventually have to allow statements about the probabilities of events, but attaching probabilities to all statements has the following objections:*

1. *It is not clear how to attach probabilities to statements containing quantifiers in a way that corresponds to the amount of conviction people have.*
2. *The information necessary to assign numerical probabilities is not ordinarily available. Therefore, a formalism that required numerical probabilities would probably be epistemologically inadequate (37).*

Supporters of probability approaches have generally responded to this problem in one of three ways: they have applied their analyses to areas in which large bodies of data on which to base prior probabilities are relatively available; they

have relied on estimates from experts in the problem domain; or they have reinterpreted their measures, usually as confidence levels or some similar concept (see below).

The practice of using estimates from experts deserves particular attention because it has raised separate difficulties of its own. People are not very good at estimating probabilities or at informal reasoning involving them. The classical study demonstrating this was published by Tversky and Kahneman in 1974 (38), and masses of data have since been collected and analyzed supporting the fact (for a collection of studies, see Ref. 39). These results raise two questions. First, if people are bad probability estimators, what does that say about the reliability of systems that start from their informal estimates? Assurances that over the very long run the systems will be able to improve on those values provide little comfort to those who care what answers the systems give meanwhile. Second, and more subtly, people in fact reason fairly effectively in situations of uncertainty. Since people appear to be bad at probability reasoning, maybe they are doing something else. If that is so, AI researchers may profit as much as psychologists from finding out what.

Furthermore, to apply statistical formulas correctly, it is usually necessary to know whether the events involved are independent. Formulas that work for independent events may provide nonsense results when applied to dependent ones, and vice versa. But in the settings in which AI systems operate, that information—whether events are mutually dependent and, if so, how they are related—is frequently lacking. Bayesians typically deal with this problem by treating all events as independent unless there are known dependencies between them. In defense of the approach, it is argued that the practice lets the system detect divergences from expectations and thereby revise the assumption (for a fuller defense against this and other charges, see Refs. 40 and 41). However, until enough anomalies have been found to bring about the appropriate revision, the system will make bad, and unfounded, predictions.

In addition to these points, the following criticisms of replacing default reasoning by probability analyses have been raised:

1. There are kinds of generalizations (see Ref. 16), including those based on notions of typicality (e.g., see, Refs. 1–3) and normic generalizations (see Refs. 42 and 43), that are not probability-based and that are crucial to the way people actually reason in many kinds of situation. Since these kinds of generalizations cannot be modeled adequately by probabilities, probability reasoning can at best supplement, and not replace, other forms of default reasoning.
2. Probability models fail to distinguish ignorance from known degrees of uncertainty and fail to capture the degree of certainty of prior probability judgments. Hence they represent and use more information than is actually known. These criticisms underlie many of the applications of the Dempster–Shafer theory (see Refs. 44 and 45).
3. Single-measure models inadequately reflect the grounds for belief (and therefore the circumstances of revision) of a proposition; this is Cohen's argument for his theory of endorsements, for instance (see Refs. 29 and 30).
4. Uses of probability theory overlook the distinction between uncertainty and vagueness; this is Zadeh's primary criticism (46–50; see also below).

5. Using the Dempster–Shafer theory, very small probabilities do not act mathematically like probabilities of zero, and so treating them as such may be hazardous. These mathematical results raise questions about both inaccuracies and thresholding (see Ref. 51).

For all these reasons the use of probability reasoning to reason in contexts of uncertainty and to measure the degree of uncertainty of beliefs remains disputed within the AI community. Where full-fledged probability analyses are possible, including among other things a well-based analysis of dependence conditions among events, the principles of classical probability theory stand unchallenged; but there seem to be no safe shortcuts, and it is far from clear that all situations of interest are amenable to the kind of analysis a full probabilistic treatment requires. These concerns have also sparked the development of other, non-probabilistic measure-based approaches to reasoning in uncertainty.

**Fuzzy Sets and Fuzzy Logic.** The fuzzy-reasoning approach, developed primarily by Zadeh (46–50), holds that classical treatments of properties, measures, and truth and falsity oversimplify reality by assuming that the fundamental questions have yes-or-no answers, when really people should be looking for to-what-extent answers. The starting point for these views is fuzzy set theory. Classical set theory holds that for any set and any individual, either that individual (absolutely) belongs to a set or (absolutely) does not. That is, you can think of a set  $S$  as determining a function  $c_S$  (usually called the characteristic function of the set) as follows:

$$c_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{if } x \notin S \end{cases}$$

For fuzzy set theory, the characteristic function (in which 1 intuitively represents “true,” or “yes, it is a member,” and 0 intuitively represents “false”) is replaced by a measure  $\mu_S$ , which has the following property that for any  $x$ ,  $0 \leq \mu_S(x) \leq 1$ . This function gives the measure of the extent to which  $x$  belongs to  $S$ . Once again, a value of 1 represents absolute membership, while a value of 0 represents absolute exclusion; but now there are permitted values in between, which intuitively represent degrees of partial membership.

Parallel to fuzzy set theory, Zadeh has proposed a fuzzy propositional logic, which assigns values between 0 and 1 to propositions instead of the more usual two truth values (true and false). These assignments were originally proposed as degrees of truth, but Zadeh now refers to them as probabilities (51). In this system it is not clear how the values of complex propositions (i.e., propositions made up of several components joined by “and,” “or,” “if–then,” etc.) are related to those of their constituents; some results on this problem are given in Ref. 52.

More recently, Zadeh has extended his views to include a theory of fuzzy numbers and fuzzy arithmetic (53), under which the measures themselves may in fact be intervals. It is these intervals as measures that he proposes as fuzzy quantifiers in dealing with syllogistic reasoning (50) and that he claims capture probabilities. However, he continues to distinguish his approach from statistical ones, partly on the grounds that his fuzzy numbers (intervals) reflect the uncertainty of the probability determination itself and partly due to claims of greater flexibility and computational efficiency. Given that the syllogistic reasoning he is dealing with is by its nature

limited to claims about the relationship between two unary predicates (i.e., two sets, which can be viewed as events if the potential members are outcomes), the combination rules he provides for fuzzy syllogistic reasoning are essentially restatements of the upper and lower probability bounds classical statistical reasoning would provide depending on how the premises of the syllogism are related to each other (one bound representing total independence, the other total dependence). For more on fuzzy reasoning, see Ref. 54.

**Other Measures.** Critics of classical probability who nonetheless want a numerical measure that somehow reflects the state of the system's knowledge about, evidence for, or attitude toward a proposition's status have been seeking alternatives for some time; one of the earliest such is the system of separately collecting and evaluating evidence for and against hypotheses motivated by the experience of developing MYCIN (see discussion in Ref. 55). A recent such proposal is Rich's suggested likelihood reasoning (56). There are essentially two problems with most of these. First, if they are not measuring probability, what are they measuring? It is not enough to give a word ("confidence," "certainty," "degree of belief," "degree of confirmation," "likelihood," etc.). There is a well-developed theory of statistics, with interpretations for the precise meaning of "probability" and means of demonstrating that the formulas presented measure something interesting. Any competing theory will have to present the same thing: not merely a name, but a clear analysis of what is being measured, how it is different from a probability, why people should nonetheless be interested in it, and how people can feel certain that the computations used to derive the numbers in fact compute anything interesting. To date, most work in the direction remains suggestive rather than persuasive.

One interesting and increasingly common approach applies Dempster-Shafer theory (44,45). This has attracted a great deal of attention, and numerous researchers are developing measure-propagation rules on its basis (e.g., see Refs. 57-59). The model that is usually applied arises out of classical statistics but differs from the classical approach in that instead of assigning numbers as probabilities, it assigns pairs representing lower and upper bounds on probability estimates. The range below the low and high elements of the pair may be viewed as measuring the amount of ignorance concerning the actual probability, i.e., the degree (or lack) of confidence in the estimate. This version of Dempster-Shafer theory is attractive for several reasons. First, it builds on classical probability theory, thus inheriting much of its theoretical foundations. Second, it seems not to overcommit by not forcing precise statements of probabilities: its probabilities do not seem to provide more information than is really available. Third, reflecting the degree of ignorance of probability estimate is attractive. Fourth, Dempster-Shafer theory provides rules for combining probabilities and thus for propagating measures through the system. This fourth point is possibly the most attractive, but it is also one of the most controversial since the propagation method is an extension of the multiplication rule for independent events. Since many of the applications involve events that are surely dependent, that rule is, by classical statistical criteria, inapplicable. The tendency to assume that events are independent unless proven otherwise has stimulated a large proportion of the criticism of probability approaches; as it stands, Dempster-Shafer theory suffers the same ill.

## Conclusions

Default reasoning is one of the most active fields of research in AI today. The approaches being taken to deal with this set of problems cover a wide range, from classical probability theory to emphatically nonclassical logics. The extent to which the solutions to the problems posed by reasoning in uncertainty and with incomplete information are solvable at all, the classes into which their solutions will fall, and the extent to which those solutions will depend on new means of reasoning as opposed to new, more efficient means of dealing with ever-increasing stores of specific knowledge all remain open to debate. The field is ripe for a clear theory of default reasoning, its bases, and its requirements.

## BIBLIOGRAPHY

1. E. Rosch, "Cognitive representations of semantic categories," *J. Exp. Psych. Gen.* **104**, 192-233 (1975).
2. E. Rosch and C. B. Mervis, "Family resemblances: Studies in the internal structure of categories," *Cog. Psych.* **7**, 573-605 (1975).
3. E. Rosch, C. B. Mervis, W. D. Gray, D. B. Johnson, and P. Boyes-Braem, "Basic objects in natural categories," *Cog. Psych.* **8**, 382-439 (1976).
4. D. McDermott and J. Doyle, "Non-monotonic logic I," *Artif. Intell.* **13**, 41-72 (1980).
5. D. McDermott, "Non-monotonic logic II," *JACM* **29**, 33-57 (1982).
6. Reference 4, p. 46.
7. R. Reiter, "A logic for default reasoning," *Artif. Intell.* **13**, 81-132 (1980).
8. R. Reiter and G. Criscuolo, On Interacting Defaults, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., Canada, pp. 270-276, 1981.
9. D. W. Etherington and R. Reiter, On Inheritance Hierarchies with Exceptions, *Proceedings of the Third National Conference on Artificial Intelligence*, Washington, DC, pp. 104-108, 1983.
10. D. S. Touretzky, Implicit Ordering of Defaults in Inheritance Systems, *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, pp. 322-325, 1984.
11. R. Reiter, On Closed World Data Bases, in H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum, New York, pp. 55-76, 1978.
12. M. Minsky, A Framework for Representing Knowledge, in P. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, pp. 211-277, 1975.
13. P. Hayes, The Logic of Frames, in D. Metzing (ed.), *Frame Conceptions and Text Understanding*, Walter de Gruyter, Berlin, FRG, pp. 46-61, 1979.
14. Reference 7, p. 82.
15. Reference 7, p. 82-83.
16. J. T. Nutter, Defaults Revisited, or "Tell me If You're Guessing," *Proceedings of the Fourth Annual Conference of the Cognitive Science Society*, Ann Arbor, MI, pp. 67-69, 1982.
17. J. T. Nutter, What Else is Wrong With Non-Monotonic Logics? Representational and Informational Shortcomings, *Proceedings of the Fifth Annual Conference of the Cognitive Science Society*, Rochester, NY, 1983.
18. Reference 7, pp. 93-94.
19. Reference 7, p. 87.
20. J. McCarthy, "Circumscription—A form of non-monotonic reasoning," *Artif. Intell.* **13**, 27-40 (1980).
21. J. McCarthy, "Addendum: Circumscription and other non-monotonic formalisms," *Artif. Intell.* **13**, 171-172 (1980).

22. J. McCarthy, Applications of Circumscription to Formalizing Commonsense Knowledge, *AAAI Workshop on Non-Monotonic Reasoning*, New Paltz, NY, pp. 295–323, 1984.
23. Reference 20, p. 27.
24. Reference 20, p. 29.
25. Reference 20, p. 37.
26. M. Davis, "The mathematics of non-monotonic reasoning," *Artif. Intell.* **13**, 73–80 (1980).
27. D. J. Israel, What's Wrong With Non-monotonic Logic? *Proceedings First Annual National Conference on Artificial Intelligence*, Stanford, CA, pp. 25–50, 1980.
28. R. C. Moore, Semantical Considerations on Nonmonotonic Logic, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 272–279, 1983.
29. P. R. Cohen and M. R. Grinberg, "A theory of heuristic reasoning about uncertainty," *AI Mag.* **4**, 17–24, (1983).
30. P. R. Cohen and M. R. Grinberg, A Framework for Heuristic Reasoning About Uncertainty, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 355–357, 1983.
31. M. Sullivan and P. R. Cohen, An Endorsement-Based Plan Recognition Program, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 475–479, 1985.
32. H. J. Levesque, A Formal Treatment of Incomplete Knowledge, Ph.D. Thesis, University of Toronto, Toronto, Ontario, 1981.
33. H. J. Levesque, The Logic of Incomplete Knowledge Bases, in M. L. Brodie, J. Mylopoulos, and J. W. Schmidt (eds.), *Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*, Springer-Verlag, New York, pp. 165–186, 1983.
34. H. J. Levesque, The Interaction with Incomplete Knowledge Bases: A Formal Treatment, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., pp. 240–245, 1981.
35. J. T. Nutter, Default Reasoning Using Monotonic Logic: A Modest Proposal, *Proceedings of the National Conference on Artificial Intelligence*, Washington, DC, pp. 297–300, 1983.
36. R. O. Duda, P. E. Hart, N. J. Nilsson, and G. L. Sutherland, Semantic Network Representations in Rule-Based Inference Systems, in D. A. Waterman and F. Hayes-Roth (eds.), *Pattern-Directed Inference Systems*, Academic Press, New York, pp. 203–223, 1978.
37. J. McCarthy and P. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 4, Edinburgh University Press, Edinburgh, pp. 463–502, 1969.
38. A. Tversky and D. Kahneman, "Judgement under uncertainty: Heuristics and biases," *Science* **185**, 1124–1131 (1974).
39. D. Kahneman, P. Slovic, and A. Tversky (eds.), *Judgement under Uncertainty: Heuristics and Biases*, Cambridge University Press, New York, 1982.
40. P. Cheeseman, In Defense of Probability, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 1002–1009, 1985.
41. M. L. Ginsberg, Does Probability Have a Place in Non-monotonic Reasoning? *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 107–110, 1985.
42. M. Scriven, Truisms as the Grounds for Historical Explanations, in P. Gardiner (ed.), *Theories of History*, Free Press, New York, pp. 443–475, 1959.
43. M. Scriven, New Issues in the Logic of Explanation, in S. Hook (ed.), *Philosophy and History*, New York University Press, New York, pp. 339–361, 1963.
44. G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ, 1976.
45. A. P. Dempster, "A generalization of Bayesian inference," *J. Roy. Stat. Soc.* **30** (Series B), 205–247 (1968).
46. L. A. Zadeh, "Fuzzy sets," *Inf. Ctrl.* **8**, 338–353 (1965).
47. L. A. Zadeh, "Fuzzy algorithms," *Inf. Ctrl.* **12**, 92–102 (1968).
48. L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems* **1**, 3–28 (1978).
49. L. A. Zadeh, Possibility Theory and Soft Data Analysis, in L. M. Cobb and R. M. Thrall (eds.), *Mathematical Frontiers of the Social and Policy Sciences*, A.A.A.S. Selected Symposium, Vol. 54, Westview, Boulder, CO, pp. 69–129, 1981.
50. L. Zadeh, Syllogistic Reasoning as a Basis for Combination of Evidence in Expert Systems, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 417–419, 1985.
51. D. Dubois and H. Prade, Combination and Propagation of Uncertainty with Belief Functions—A Reexamination, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 111–113, 1985.
52. A. R. Aronson, B. E. Jacobs, and J. F. Minker, "A note on fuzzy deduction," *JACM* **21**, 599–603 (1980).
53. A. Kaufmann and M. M. Gupta, *Introduction to Fuzzy Arithmetic*, Van Nostrand Reinhold, New York, 1985.
54. E. H. Mamdani and B. R. Gaines, *Fuzzy Reasoning and its Applications*, Academic, London, 1981.
55. B. B. Buchanan and E. H. Shortliffe, *Rule Based Expert Systems*, Addison-Wesley, Reading, MA, 1984.
56. E. Rich, Default Reasoning as Likelihood Reasoning, *Proceedings of the Third National Conference on Artificial Intelligence*, Washington, DC, pp. 348–351, 1983.
57. M. L. Ginsberg, Non-Monotonic Reasoning Using Dempster's Rule, *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, pp. 126–129, 1984.
58. T. M. Strat, Continuous Belief Functions for Evidential Reasoning, *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, pp. 308–313, 1984.
59. S. Y. Yu and H. E. Stephanou, A Set-Theoretic Framework for the Processing of Uncertain Knowledge, *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, pp. 216–221, 1984.

### General References

- P. Besnard, R. Quiniou, and P. Quinton, A Theorem-Prover for a Decidable Subset of Default Logic, *Proceedings of the Third National Conference on Artificial Intelligence*, Washington, DC, pp. 27–30, 1983.
- G. W. Cottrell, Parallelism in Inheritance Hierarchies with Exceptions, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 194–202, 1985.
- P. Gärdenfors, "Qualitative probability as an intensional logic," *J. Phil. Log.* **4**, 171–185 (1975).
- J. W. Goodwin, A Process Theory of Non-monotonic Inference, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 185–187, 1985.
- T. Imielinski, Results on Translating Defaults to Circumscription, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 114–120, 1985.
- R. C. T. Lee, "Fuzzy logic and the resolution principle," *JACM* **19**, 109–119, (1972).
- V. Lifschitz, Computing Circumscription, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 121–127, 1985.

- W. Lukasiewicz, General Approach to Non-Monotonic Logics, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Washington, DC, pp. 352–354, 1983.
- W. Lukasiewicz, Two Results on Default Logic, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 459–461, 1985.
- C.-R. Rollinger, How to Represent Evidence: Aspects of Uncertain Reasoning, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 358–361, 1983.
- E. Sandewall, An Approach to the Frame Problem, and its Implementation, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 7, J Wiley, New York, pp. 195–204, 1972.
- E. Sandewall, A Functional Approach to Non-Monotonic Logic, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 100–106, 1985.
- E. Shapiro, Logic Programs with Uncertainties: A Tool for Implementing Rule-Based Systems, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 529–532, 1983.
- R. Weyrauch, "Prolegomena to a theory of mechanized formal reasoning," *Artif. Intell.* 13, 133–170 (1980).
- T. Winograd, "Extended inference modes in reasoning by computer systems," *Artif. Intell.* 13, 5–26, (1980).

J. T. NUTTER  
Virginia Tech

## REASONING, FOCUS OF ATTENTION

The topic of reasoning in AI is broad. The goal of designing algorithms and programs to emulate human reasoning makes up a large portion of AI research. One of the major drawbacks to achieving this task is due to resource limitations. Nearly all AI programs suffer to some extent from resource limitations. Computer programs in general, and AI programs in particular, require various resources. These resources include the solution spaces that the problems lie in, the processing time needed to search these solution spaces, and the characterization of the solution spaces. A solution space can be viewed as the set of all possible states that can be generated from a given state. One view of reasoning is the process of exploring the solution space.

One of the earliest domains to attract work in AI was game playing (qv) in general and chess playing in particular (see Computer chess methods). One strategy in designing a chess-playing program is to generate all possible moves from a given board position. From each of these moves generate all possible countermoves; from each of those moves generate all possible responses; and so on, until the solution is derived. This method, known as the brute-force method, suffers from a major resource limitation, namely, that the time it takes to search all possible moves and countermoves is prohibitive. There have been numerous techniques developed for searching/exploring these large solutions spaces, and a major branch of AI research, known simply as search (qv), has gone into developing these techniques.

### Formal Logic

In the domain of reasoning using formal logic (qv), constraint may be placed on the number of inferences that are made. One

example of using formal logic is in the area of problem solving (qv). A database may exist that contains assertions such as "All dogs are mammals," "All mammals have hair," and "Fido is a dog." The number of inferences needed to conclude that Fido has hair is greater than the number of inferences needed to conclude that Fido is a dog. If resources are limited, it may be desirable to limit the number of inferences that can be made. This is useful in the cases where there are either several items that need to be proved or several ways to prove one item. Two general approaches exist. One is to suspend a currently running reasoning chain after  $n$  inferences, with the option of resuming that process in the current state at a later time. This type of process is called a coroutine (qv). The second approach is to keep a list or agenda of the tasks that remain ordered, based on some sort of priority system, and run each one for a fixed number of instances. See Moore (1) for a logic-based-reasoning system, KRL (qv) (2) is a language that supplies the programmer with the ability to form these agendas. See Lenat's AM (qv) (3) program for the use of an agenda to control inferences in the domain of discovering mathematical theorems.

### Natural Language

Natural-language processing systems have had to deal with resource limitations (see Natural-language understanding). The problem can be stated as "How can a natural-language system be written without encoding the entire knowledge of the world?" Solutions diversify from constraint of the domains such that only a limited amount of world knowledge is encoded. Often this can be done in natural-language front-end systems (generally to databases). Microworlds can be developed where, in these cases, the entire world can be encoded. Reasoning about texts often leads to inferring the meanings that are communicated. Trying to make all possible inferences can lead to combinatorial explosion. Analyzing all such inferences in order to determine which is most appropriate for a given situation or problem is also prohibitive. Frames and scripts were developed to constrain the amount of plausible inferences that are made in a given context.

One of the most general mechanisms for dealing with resource-limited reasoning is to establish a focus of attention. This approach confines the processing to a limited region at a particular instance. Although the focus of attention may shift as the processing proceeds, there should always be a current focus of attention. The work by Grosz (4) on defining the characteristics of the focus of attention has mainly concentrated on its relationship to dialogue. Next, two other systems are briefly examined that use the concept of focus in order to handle resource limitations.

The HEARSAY II system (qv), designed to interpret spoken natural language, tried to devise a focus-of-attention mechanism to handle the information-explosion problem. Erman et al. (5), in describing the HEARSAY II system, equate the problem of focus of attention with that of resource allocation. Autonomous processes, sometimes called knowledge sources (KS) or demons, compete with each other for control. They proposed five fundamental principles for handling resource-limitation problems arising from management of these independent processes (5).

1. *The competition principle:* The best of several local alternatives should be performed first.



2. *The validity principle*: KSs operating on the most valid data should be executed first.
3. *The significance principle*: Those KSs whose response frames are most important should be executed first.
4. *The efficiency principle*: Those KSs that perform most reliably and inexpensively should be executed first.
5. *The goal-satisfaction principle*: Those KSs whose responses are most likely to satisfy processing goals should be executed first.

The above is an example of a heuristically guided method for dealing with resource limitations.

### Expert Systems

The area of expert systems has had to grapple with the problem of resource-limited reasoning. Many expert systems (qv) are written in the form of production systems. Reasoning is accomplished by firing production rules until no more rules are applicable. The final state is the result. The production rules that were fired and lead from the start state to the solution state make up the reasoning chain. The traversal of this chain can be extremely time-consuming, considering that there may be hundreds of production rules that make up the links in this reasoning chain. Chandrasekaran and Mittal (6) suggest compiling rules and reorganizing them under concepts in a hierarchical structure. It is this hierarchy that imposes a focus on various rules and their relationship to other rules. This has the effect of short-circuiting long inference chains in either commonly used or conceptually linked fashion.

The above are samples of the many AI systems and/or theories that explicitly attempt to deal with resource-limited reasoning and in particular its relationship to the focus of attention. Virtually all AI programs must confront the issue either explicitly or as is more often the case, implicitly.

### BIBLIOGRAPHY

1. R. C. Moore, *Reasoning from Incomplete Knowledge in a Procedural Deduction System*, Garland, New York, 1980.
2. D. G. Bobrow and T. Winograd, "An overview of KRL, a knowledge representation language," *Cogn. Sci.* 1(1), 3-46 (1977).
3. D. B. Lenat, On Automated Scientific Theory Formation: A Case Study Using the AM Program, in J. E. Hayes, D. Michie, and L. I. Mikulich (ed.), *Machine intelligence*, Vol. 9, Halsted, a division of Wiley, New York, pp. 251-286, 1977.
4. B. J. Grosz, Focusing and Description in Natural Language Dialogues, in A. Joshi, B. Webber, and I. Sag (ed.), *Elements of Discourse Understanding*, Cambridge University Press, Cambridge, UK, pp. 84-105, 1981.
5. L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy "The HEARSAY-II speech understanding system: Integrating knowledge to resolve uncertainty," *Comput. Surv.* 12(2), 213-253 (1980); and *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, p. 35, 1977.
6. Chandrasekaran and S. Mittal, "Deep versus compiled knowledge approaches to diagnostic problem-solving," *Int. J. Man-Machine Stud.* (Special Issue: Developments in Expert Systems) 20(1), 425-436 (November 1983).

J. ROSENBERG  
SUNY at Buffalo

## REASONING, NONMONOTONIC

In many situations appropriate reasoning involves drawing "default" conclusions based on incomplete data. The results of such reasoning are in general unsound, i.e., they are not necessarily true even though the data on which they are based may be true. Nonetheless, such reasoning can play an important role. For instance, a robot may assume that in the absence of information to the contrary, its arms may be used to transport objects in the "normal" manner. Similarly, in normal conversation the appropriate response to "Did you understand that article?" is not "I understand 90% of it" if in fact the latter speaker understood it all. That is, even though the indicated response is not strictly false, it is misleading because it encourages the questioner to conclude (unsoundly) that only 90% of the article was understood. These and other examples strongly suggest that reasoning by default (see Reasoning, default) is prevalent throughout commonsense reasoning (qv).

Marvin Minsky (1) coined the term "nonmonotonic logic" in developing an argument that tools of formal logic are inadequate to the task of representing commonsense reasoning. His argument involved an analysis of the use of defaults. As indicated above, these are conclusions based on the absence of contrary information. Consider the example of concluding from the knowledge that Tweety is a bird that Tweety can fly. Such a conclusion is not necessarily correct, although it certainly can be very useful in some situations. One could try to specify precisely those situations in which such a conclusion is sound, but any effort to do so quickly leads to despair. The possible circumstances in which any presumed correct line of reasoning can be defeated astounds: Tweety may be an ostrich, may have a broken wing, may be chained to a perch, may be too weak, etc. Indeed, the problem is virtually the same as that of the well-known frame problem: The special conditions relevant to determining what may be the case in a complex environment defies precise specification. Although there appears to be a meaningful sense of certain "typical" situations (such as "typical" birds being ones that, among other things, can fly), it is notoriously hard to define typicality.

However, Minsky's claim was more than this: He argued that, first, conclusions of the sort given in the Tweety example are contingent on what else is known (e.g., if it is already known that Tweety cannot fly, one refrains from concluding the opposite) and, second, such conclusions do not obey the customary phenomenon of "monotonicity" of formal systems of logic. That is, a standard logic  $L$  has the property that if  $\phi$  is a theorem of  $L$  and if  $L$  is augmented to  $L^*$  by additional axioms, then  $\phi$  remains a theorem of  $L^*$ . Indeed, the same proof of  $\phi$  in  $L$  is a proof of  $\phi$  in  $L^*$ . However, commonsense reasoning seems to allow a "proof" (at least in the form of a tentative supposition) that Tweety can fly, given only that Tweety is a bird, whereas in the augmented state in which it is known also that Tweety cannot fly, no such "proof" is forthcoming. In effect, account seems to be taken of what the reasoner does not know, an issue already much studied in the area of databases in the context of the closed-world assumption (2), in which any atomic formula not explicitly present in the database is intended (or assumed) to be false. Similarly, inheritance hierarchies (qv) provide mimicked traits (e.g., flying) for subclasses (e.g., robins) of other classes (e.g., birds) unless made exceptional (e.g., ostriches).

It is true that any straightforward attempt to represent such reasoning in terms of sentences in a traditional "mono-

tonic" logic in which the stated conclusions are theorems will fail for the simple reason that these logics necessarily have the original theorem (Tweety can fly) carried over to the augmented theories by virtue of their monotonicity. Several questions then arise:

1. Are there other formal logics that can represent such reasoning?
2. Has commonsense reasoning been fairly portrayed here or are there other factors involved that might change the assessment of the role of nonmonotonicity?
3. Might not a clever use of monotonic logic allow the effect of nonmonotonic deductions?

Minsky seems to have concluded that formal methods per se are inappropriate to capture such reasoning, whereas others have taken his ideas as a challenge by which to find more powerful formal methods. Out of this challenge has arisen a substantial field of research in nonmonotonic reasoning.

In fact, vigorous efforts have been made toward answering each of the three questions above, and the terrain has by now shown itself to be a rich and varied one involving ideas from diverse parts of artificial intelligence, logic, natural language, and philosophy. One theme that seems to have emerged is that a key element in commonsense reasoning dealing with uncertainty (due to the abundance of special conditions defying specification) is self-reference (see Self-reference): The reasoning entity utilizes information about the extent of its own knowledge. Indeed, answers suggested to the above three questions can be viewed in terms of their approach to representing such self-reference. This is explored in what follows.

### Nonmonotonic Formalisms

Two distinct formalisms emerged around 1980 that attempted to capture the essence of nonmonotonic reasoning by providing a new kind of logical framework. One, due to McDermott and Doyle (3), bears simply the name of "nonmonotonic logic," and the other, due to Reiter (4), is called "default logic." Both employ inferential tools, making explicit use of information about what information the formalism itself has available to it. In both cases new syntactic and inferential constructs are developed. Each of these are discussed in turn.

**Nonmonotonic Logic of McDermott and Doyle.** The nonmonotonic logic of McDermott and Doyle (3) (NML for short) takes as point of departure the desire to represent axiomatically such notions as "If an animal is a bird, then unless proven otherwise, it can fly." To do this they introduce into the language (initially a first-order language) a modal operator  $M$ , so that if  $p$  is a formula then so is  $Mp$  (read " $p$  is consistent"). Now in this language it is possible to write formulas that seem to express the kind of reasoning given earlier. For instance, the formula

$$(x)[\text{Bird}(x) \ \& \ M\text{Flies}(x) \rightarrow \text{Flies}(x)]$$

appears to convey information appropriate to concluding of typical birds that they can fly. A means is needed to characterize deductions with formulas containing the operator  $M$ , however, and this McDermott and Doyle go to some length to develop. As this is essential to their treatment, some time is spent examining it now. To provide an example for the follow-

ing discussion, let  $A$  be the theory  $\{\text{Bird}(x) \ \& \ M\text{Flies}(x) \rightarrow \text{Flies}(x), \text{Bird}(\text{Tweety})\}$ .

At first blush, it would appear easy to state what is wanted. For if indeed the formula  $p$  is consistent (with the rest of the axioms of the particular instance of NML one wishes to utilize), and if in "typical" situations (i.e., ones in which  $p$  is consistent) the formula  $q$  happens to be true, a rule such as "from  $Mp$  deduce  $q$ " seems appropriate. However, McDermott and Doyle have chosen  $M$  to be a part of the language itself, i.e.,  $Mp$  is a formula as well as  $p$ . This means that a mechanism is needed to make it possible to prove formulas such as  $Mp$ , and this is problematic since proofs of consistency are not only notoriously hard in general but in fact are usually impossible within the same axiomatic system with respect to which consistency is sought. To deal with this problem, McDermott and Doyle extend the notion of proof to allow a kind of consistency test at the expense of effectiveness. (In fact, all formal approaches to nonmonotonic reasoning seem to run into this same issue.) Their notion of proof is as follows: If  $L$  is a first-order language modified by the addition of a modal operator  $M$ ,  $A$  is a theory in the language  $L$ , and  $S$  is a set of formulas in the language  $L$  of  $A$ , let

$$NM_A(S) = \text{Th}(A \cup \text{As}_A(S))$$

where

$$\text{As}_A(S) = \{Mq : q \in L \text{ and } \neg q \notin S\} - \text{Th}(A)$$

Here  $\text{Th}(A)$  is the usual set of first-order consequences of  $A$ , and  $\text{As}_A(S)$ , the so-called set of assumptions from  $S$ , consists of those formulas  $Mq$  not in  $\text{Th}(A)$  for which  $\neg q$  is not in  $S$ . Intuitively, an  $Mq$  that is not already proven is to be considered an assumption on the basis of  $S$  if  $S$  does not rule  $q$  out, i.e.,  $Q$  is considered to be "possible." The idea is to adjoin assumptions to  $A$  and find all (usual) consequences, this producing the set  $NM_A(S)$ . Of course,  $S$  could be  $A$  itself, or even empty. However, when  $NM_A$  is formed, new formulas are thereby available for use (i.e., they are considered "proven"), and these may themselves provide the basis for another round of assumptions. So  $S$  plays the role of a recursion variable, and a fixed point of  $NM_A(S)$  is sought. It is desired then to consider as theorems precisely those formulas contained in all fixed points  $S$ . However, some theories  $A$  have no such fixed points, and for these such a definition will not do. McDermott and Doyle settle on the entire language  $L$  in such cases, thereby defining the set of theorems nonmonotonically derivable from  $A$  as

$$\text{TH}(A) = \cap(\{L\} \cup \{S : NM_A(S) = S\})$$

In terms of the example theory  $\{\text{Bird}(x) \ \& \ M\text{Flies}(x) \rightarrow \text{Flies}(x), \text{Bird}(\text{Tweety})\}$ , every fixed point  $S$  contains the sentence  $\text{Flies}(\text{Tweety})$ . Intuitively, since  $\neg \text{Flies}(\text{Tweety})$  is not initially in the theory, and each stage of generating new assumptions produces only additional sentences such as  $M\text{Flies}(\text{Tweety})$  as well as their ordinary consequences [such as  $\text{Flies}(\text{Tweety})$ ], then  $\text{Flies}(\text{Tweety})$  remains present in all iterations of the assumption process. Thus,  $\text{Flies}(\text{Tweety})$  will be a nonmonotonic theorem of  $A$ .

Note that any attempt to calculate  $\text{TH}(A)$  leads to consistency tests. For in iterating  $NM_A(S)$  for  $S$  initially empty, one arrives immediately at the necessity of determining whether, for any given  $q$ ,  $Mq$  is in  $\text{Th}(A)$ . But this is in general undecidable, and amounts precisely to determining whether  $A \cup$

$\{\neg Mq\}$  is inconsistent. McDermott and Doyle acknowledge this difficulty and show that in very restricted cases—essentially propositional logic—there is a remedy. [They also define a notion of model for NML; however, there is some dispute as to the completeness of their definition (5).]

McDermott (6) tries to strengthen NML so as to overcome certain weaknesses in the original version, in particular the fact that  $Mp$  and  $\neg p$  are not contradictory. The new effort makes fuller use of the modal character of the language, but in the most interesting case it collapses into equivalence with an ordinary monotonic logic.

**A Distinction Due to Moore.** Moore (7) reexamines the underlying goals of NML and concludes that two ideas are being conflated: typicality on the one hand and beliefs about one's beliefs on the other. He distinguishes between concluding Tweety can fly on the basis that it is not known that Tweety cannot fly and that typically birds can fly, and concluding Tweety can fly on the basis that it is not known that Tweety cannot fly and that "I would know it if Tweety couldn't fly." Moore argues that the former is intended to be approximate and error-prone, and the latter (which he calls "autoepistemic reasoning") is intended to be sound. He devises a consistent logic for the latter form of reasoning.

It does appear that autoepistemic reasoning forms a part of commonsense reasoning. The example above is not as striking as one given by Moore: "I would know it if I had an elder brother." Here one is presumably not merely stating a belief about typicality (that one typically knows one's older brothers, although that seems true enough) but rather a belief that "I" specifically do know of all "my" brothers. Admittedly this is arguable since one can think of situations in which an older brother may be unknown, but they are not likely to be taken seriously, so that again a kind of typicality may be present here.

Moore points out that in autoepistemic beliefs there is a possibility of failure, i.e., the belief can be false (I may have an elder brother after all) in which case I must alter that belief, whereas in the case of typicality I may merely conclude that I am atypical regarding knowledge of brothers and yet preserve the belief that typically elder brothers are known. Still, if I do discover to my surprise that such a brother exists, it would seem likely that I would conclude immediately that I was wrong about my autoepistemic belief but that the belief still applies to most people. That is, there seems a very fine and tenuous line between the two forms of beliefs. It seems possible to move back and forth between explicit typicality beliefs in which one acknowledges their uncertainty and more stubborn autoepistemic ones, for the same assertions, depending on context, and one's willingness to alter his/her position when challenged may attest to an implicit default character even in autoepistemic cases.

It is of interest that both forms of reasoning, however, like all other nonmonotonic formalisms, depend at least implicitly on a determination that in fact certain formulas are not theorems of the formalism in question. Note that in Moore's example I must somehow determine that in fact I do not know of an elder brother before using the autoepistemic belief and *modus ponens* to conclude I have no such brother. Again, this self-referential or consistency aspect of the reasoning seems the most striking characteristic and the one presenting the greatest formal difficulty.

**Default Logic of Reiter.** Reiter (4) introduces a logic for default reasoning (denoted as DL). In specifically singling out default reasoning, Reiter identifies his concern as that of studying typicality rather than other possible nonmonotonic forms of reasoning. His formalism in fact bears close resemblance to NML, the most obvious difference being that the language is strictly first-order, with the operator  $M$  playing a role only in rules of inference rather than in axioms. Specifically, Reiter allows inference rules ("default rules") such as

$$\frac{\text{Bird}(x): \text{MFlies}(x)}{\text{Flies}(x)}$$

where  $\text{MFlies}(x)$  is intended not as an antecedent theorem to the consequent  $\text{Flies}(x)$  but instead as a condition that must be met before  $\text{Flies}(x)$  can be concluded from  $\text{Bird}(x)$ . The condition is, roughly (and as in all nonmonotonic formalisms), that  $\text{Flies}(x)$  be consistent with the rest of the axiomatic framework. Thus, if the above rule and the axiom  $\text{Bird}(\text{Tweety})$  are employed, the conclusion  $\text{Flies}(\text{Tweety})$  results. As with NML, formalizing the notion of consistency for the indicated purpose requires care. Making this precise and showing it to be useful is the bulk of the task Reiter undertakes. He employs a hierarchy of iterations along lines similar to that of NML, also arriving at a fixed point, in determining a notion of proof for default rules. Since DL uses rules in place of the axioms of NML, it would appear that in general DL is weaker than NML. This may in fact be a reason to prefer DL to NML in that one of the hoped-for features of reasoning about typicality is that limitations are placed on what conclusions are drawn. However, at present the outlines of what a reasoning system should do regarding typicality are so vague that it is difficult to defend strong claims.

Reiter and Criscuolo (8) also consider what they call interacting defaults, i.e., default rules that separately might lead to opposed conclusions, such as in "Richard Nixon is a Quaker and a Republican" where it is known, say, that typically Quakers are pacifists and Republicans are not. This appears to be a substantial difficulty for any form of nonmonotonic reasoning that pretends to deal with typicality.

### Factors Underlying Nonmonotonic Reasoning

Kowalski (9) and Israel (10) suggest that something is missing from Minsky's account of commonsense reasoning under uncertainty: The reasoning entity creating the "proof" (say that Tweety flies) must know that it does not know certain facts (such as that Tweety is an ostrich) and, further, that when this knowledge of self is properly represented, the reasoning is no longer nonmonotonic, thereby rendering unnecessary the development of new (nonmonotonic) logics. Their argument is as follows: A "default" rule such as "if  $X$  is not known then  $Y$ " (e.g., if Tweety doesn't fly is not known, then Tweety flies) is at least implicit in nonmonotonic "proofs," and so the reasoning must make use, in some fashion, of  $X$  not being known before concluding  $Y$ . This means there must be an additional mechanism  $M$  to determine that in fact  $X$  is not known. But then if the system is augmented by coming to know  $X$  (Tweety is an ostrich and therefore cannot fly, say), the system can no longer derive " $X$  is not known" as long as the mechanism  $M$  for such derivations is faithful to the facts. That is, not only has the system been augmented by now knowing  $X$ , it has had an old piece of knowledge removed (and properly so, for it no longer is

true), namely that "X is not known," and that piece of information was precisely what previously allowed the now inadmissible conclusion Y.

What has happened in such a scenario is that one logic has been replaced by another that contains additional information but also has lost some information (namely information that no longer is true because of the very presence of the new information). In effect, a reasoning system that is to know about its own reasoning would appear to require temporal changes reflecting the fact that its previous states obeyed different truths. If a system first does not know X and then later does, it was true at first that "X is not known" and later this is false. If the system itself is to have this knowledge represented (as Kowalski and Israel argue), Minsky's argument for nonmonotonicity fails, for one is no longer dealing with a strict augmentation of the original axioms. Israel in particular argues that a sequence of logics is a better way to view the situation, in which axioms are constantly added and subtracted as new facts become known, and that this process is not one of deduction but of interaction with the happenstance environment. To a certain extent, this acknowledges Minsky's point that logic is not (entirely) what is involved here. However, the insight provided by making explicit the default knowledge and mechanism M that utilizes it suggests that logic still may do all one would want of it and that other processes invoke the necessary self-referential inspections to determine whether X is still known. In fact, just such an approach is undertaken in an experimental reasoning system studied by Perlis (11).

### Remaining within First-Order Logic

McCarthy (12) has devised an ingenious means for representing and calculating knowledge about situations involving minimization of particular notions. He calls this technique "circumscription (qv)." It is noteworthy in the present context because it seems able to handle many of the kinds of reasoning with self-reference found in nonmonotonic approaches and yet stays within first-order logic. McCarthy manages this by use of an axiom schema that partially captures the notion of a model of a set of sentences, similar to (and in fact generalizing) the familiar manner of defining the natural numbers by a minimizing schema applied to the successor operation. Much work has followed his original paper. In particular, by introducing a predicate for "abnormal" (13), McCarthy has been able to capture some of the intuitions about reasoning about typicality; that is, circumscribing that predicate can lead to conclusions to the effect that Tweety can fly since it is abnormal for a bird not to fly and since abnormality is (intended to be) minimized by circumscription. Some positive and negative results on this are in Refs. 14 and 15.

### Applications and Related Work

As with much of commonsense reasoning techniques, formal nonmonotonic modes of reasoning naturally present themselves as candidates for a reasoning mechanism that could in principle be used in an intelligent robot, e.g., in conjunction with a theorem prover. So far, little has been done in a concrete way to address these issues. Some preliminary work is presented in Ref. 11. The bibliography presents numerous ref-

erences to the literature, including applications of nonmonotonic reasoning and relationships to other areas.

### BIBLIOGRAPHY

1. M. Minsky, A Framework for Representing Knowledge, in P. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.
2. R. Reiter, On Closed World Databases, in H. Gallaire and J. Minker (eds.), *Logic and Databases*, Plenum, New York, pp. 55-76, 1978.
3. D. McDermott and J. Doyle, "Non-monotonic logic I," *Artif. Intell.* **13**(1,2), 41-72 (1980).
4. R. Reiter, "A logic for default reasoning," *Artif. Intell.* **13**(1-2), 81-132 (1980).
5. M. Davis, "The mathematics of non-monotonic reasoning," *Artif. Intell.* **13**(1-2), 73-80 (1980).
6. D. McDermott, "Non-monotonic logic II: Non-monotonic modal theories," *JACM* **29**(1), 33-57 (1982).
7. R. Moore, Semantical Considerations on Non-monotonic Logic, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 272-279, 1983.
8. R. Reiter and G. Criscuolo, On Interacting Defaults, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC, pp. 270-276, 1981.
9. R. Kowalski, *Logic for Problem Solving*, North-Holland, New York, 1979.
10. D. Israel, What's Wrong with Non-monotonic Logic? *Proceedings of the First AAAI*, Stanford, CA, pp. 99-101, 1980.
11. D. Perlis, Non-monotonicity and Real-Time Reasoning, *Workshop on Nonmonotonic Reasoning*, New Paltz, NY, sponsored by AAAI, October 17-19, 1984.
12. J. McCarthy, "Circumscription: A form of non-monotonic reasoning," *Artif. Intell.* **13**(1,2), 27-39 (1980).
13. J. McCarthy, "Applications of Circumscription to Formalizing Common-Sense Knowledge," *Artif. Intell.* **28**(1), 89-116 (1986).
14. D. Etherington, R. Mercer, and R. Reiter, "On the adequacy of predicate circumscription for closed-world reasoning," *J. Computat. Intell.*, **1**(11), 11-15 (1985).
15. D. Perlis and J. Minker, "Completeness results for circumscription," *Artif. Intell.* **28**(1), 29-42 (1986).

### General References

#### A. Introductory Sources

- E. Charniak and D. McDermott, *Introduction to Artificial Intelligence*, Addison-Wesley, Reading, MA, 1985.
- E. Rich, *Artificial Intelligence*, McGraw-Hill, New York, 1983.
- P. Winston, *Artificial Intelligence*, 2nd ed., Addison-Wesley, Reading, MA, 1984.

#### B. Technical References

- Ph. Besnard, R. Quiniou, and P. Quinton, A Theorem-Prover for a Decidable Subset of Default Logic, *Proc. of the Third AAAI*, Washington, D.C., pp. 27-30, 1983.
- G. Bossu and P. Siegel, "Saturation, nonmonotonic reasoning, and the closed-world assumption," *Artif. Intell.* **25**(1), 13-63 (1985).
- K. Clark, Negation as Failure, in H. Gallaire and J. Minker (eds.), *Logic and Databases*, Plenum, New York, pp. 293-322, 1978.
- A. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *Ann. Math. Stat.* **38**, 325-339 (1967).

- J. Doyle, Truth Maintenance Systems for Problem Solving, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, p. 247, 1977.
- J. Doyle, A Glimpse of Truth Maintenance, *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, pp. 232–237, 1979.
- J. Doyle, "A truth maintenance system," *Artif. Intell.* 12, 231–272 (1979).
- J. Doyle, The Ins and Outs of Reason Maintenance, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 349–351, 1983.
- D. Etherington, Finite Default Theories, MSc Thesis, University of British Columbia, Vancouver, B.C., 1982.
- D. Etherington and R. Reiter, On Inheritance Hierarchies with Exceptions, *Proceedings of the Third AAAI*, Washington, D.C., pp. 104–108, 1983.
- S. Fahlman, D. Touretzky and W. Van Roggen, Cancellation in a Parallel Semantic Network, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC, pp. 257–263, 1981.
- D. Gabbay, Intuitionistic Basis for Non-monotonic Logic, *Lecture Notes in Computer Science*, Vol. 139, Springer, 1982.
- J. Grant and J. Minker, Answering Queries in Indefinite Databases and the Null Value Problem, Technical Report 1374, University of Maryland, 1984.
- B. Grosz, Default Reasoning as Circumscription, *Proceedings of the Workshop on Nonmonotonic Reasoning*, New Paltz, NY, sponsored by AAAI, October 17–19, 1984.
- A. Haas, Reasoning about Deduction with Unknown Constants, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC, 382–384, 1981.
- K. Konolige, Circumscriptive Ignorance, *Proceedings of the Second AAAI*, Pittsburgh, PA, pp. 202–204, 1982.
- K. Konolige, Belief and Incompleteness, SRI Technical Note 319.
- R. Kowalski, Logic for Data Description, in H. Gallaire and J. Minker (eds.), *Logic and Databases*, Plenum, New York, pp. 77–103, 1978.
- I. Kramosil, A Note on Deduction Rules with Negative Premises, *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, pp. 53–56, 1975.
- D. Kueker, Another Failure of Completeness for Circumscription, *Week on Logic and Artificial Intelligence*, University of Maryland, College Park, MD, October 22–26, 1984, unpublished manuscript.
- H. Levesque, "Incompleteness in knowledge bases," *SIGART Newslett.* 74, 150ff (1981a).
- H. Levesque, The Interaction with Incomplete Knowledge Bases: A Formal Treatment, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC, pp. 240–245, 1981.
- H. J. Levesque, A Formal Treatment of Incomplete Knowledge, Fairchild Lab for AI Research, Technical Report 3.
- V. Lifschitz, Some Results on Circumscription, *Proceedings of the Workshop on Nonmonotonic Reasoning*, New Paltz, NY, sponsored by AAAI, October 17–19, 1984.
- W. Lipski, On the Logic of Incomplete Information, *Lecture Notes in Computer Science*, Vol. 53, Springer, New York, pp. 374–381, 1977.
- W. Lukasiewicz, General Approach to Nonmonotonic Logics, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 352–354, 1983.
- J. McCarthy, "Addendum: Circumscription and other non-monotonic formalisms," *Artif. Intell.* 13(1,2), 171–172 (1980).
- J. McCarthy and P. Hayes, Some philosophical Problems from the Standpoint of Artificial Intelligence, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 4, Edinburgh University Press, Edinburgh, 1969.
- J. Martins and S. Shapiro, Reasoning in Multiple Belief Spaces, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 370–373, 1983.
- J. Minker, On Indefinite Databases and the Closed-World Assumption, *Lecture Notes in Computer Science*, Vol. 138, pp. 292–308, Springer, New York, 1982.
- J. Minker and D. Perlis, Protected Circumscription, *Proceedings of the Workshop on Nonmonotonic Reasoning*, New Paltz, NY, sponsored by AAAI, October 17–19, 1984.
- J. Minker and D. Perlis, Applications of Protected Circumscription, *Lecture Notes in Computer Science*, Vol. 170, pp. 414–425, Springer, 1984.
- R. Moore, Reasoning from Incomplete Knowledge in a Procedural Deduction System, MIT AI Lab Memo 347, Cambridge, MA, 1975.
- J. Nutter, Default Reasoning in AI Systems, MSc Thesis, SUNY at Buffalo, Computer Science Technical Report 204, 1983.
- J. Nutter, Default Reasoning Using Monotonic Logic: A Modest Proposal, *Proceedings of the Third AAAI Conference*, Washington, D.C., pp. 297–300, 1983.
- J. Nutter, What Else Is Wrong with Nonmonotonic Logics? Representational and Informational Shortcomings, *Proceedings of the Fifth Cognitive Science Conference*, Rochester, NY, 1983.
- M. A. Papalaskaris and A. Bundy, Topics for Circumscription, *Proceedings of the Workshop on Nonmonotonic Reasoning*, New Paltz, NY, sponsored by AAAI, October 17–19, 1984.
- R. Reiter, "Equality and domain closure in first-order databases," *JACM* 27(2), 235–249 (1980).
- R. Reiter, Circumscription Implies Predicate Completion (Sometimes), *Proceedings of the Second AAAI Conference*, Pittsburgh, PA, pp. 418–420, 1982.
- R. Reiter and G. Criscuolo, "Some representational issues in default reasoning," *Int. J. Comput. Math.* (1983) (special issue on computational linguistics).
- N. Rescher, Plausible Inference, Van Gorcum, Assen, The Netherlands, 1976.
- E. Rich, Default Reasoning as Likelihood Reasoning, *Proceedings of the Third AAAI Conference*, Washington, D.C., 1983.
- E. Sandewall, "An approach to the frame problem and its implementation," *Mach. Intell.* 7 (1972).
- E. Sandewall, Partial Models, Attribute Propagation Systems, and Non-monotonic Semantics, Linköping University, LITH-IDA-R-83-01, 1983.
- G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, Cambridge, MA, 1976.
- R. Stalnaker, A Note on Non-monotonic Modal Logic, Department of Philosophy, Cornell University, 1980.
- D. Touretzky, Implicit Ordering of Defaults in Inheritance Systems, *Proceedings of the Fourth AAAI Conference*, Austin, TX, 1984.
- D. Touretzky, The Mathematics of Inheritance Systems, Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, PA, 1984.
- S. Vere, "Multi-level counterfactuals for generalizations of relational concepts and productions," *Artif. Intell.* 14(2), 139–164 (1980).
- R. Weyhrauch, "Prolegomena to a theory of mechanized formal reasoning," *Artif. Intell.* 13(1-2), 133–170 (1980).
- T. Winograd, "Extended inference modes in reasoning by computer systems," *Artif. Intell.* 13(1-2), 5–26 (1980).
- L. Zadeh, Fuzzy Sets and Information Granularity, in M. Gupta, R. Ragade, and R. Yager (eds.), *Advances in Fuzzy Set Theory*, North-Holland, Amsterdam, pp. 3–18, 1979.

D. PERLIS  
University of Maryland

## REASONING, PLAUSIBLE

### Coping with Uncertainty in Expert Systems

The management of uncertainty in expert systems (qv) has usually been left to ad-hoc representations and combining rules lacking either a sound theory or clear semantics. However, the aggregation of uncertain information (facts) is a recurrent need in the reasoning process of an expert system. Facts must be aggregated to determine the degree to which the premise of a given rule has been satisfied, to verify the extent to which external constraints have been met, to propagate the amount of uncertainty through the triggering of a given rule, to summarize the findings provided by various rules or knowledge sources or experts, to detect possible inconsistencies among the various sources, and to rank different alternatives or different goals.

There is no uniformly accepted approach to solve this issue. A variety of approaches are described as part of the review of the state of the art in reasoning with uncertainty.

### Sources of Uncertainty

In a recent survey of reasoning with uncertainty (1) it is noted that the presence of uncertainty in reasoning systems is caused by a variety of sources: the reliability of the information, the inherent imprecision of the representation language in which the information is conveyed, the incompleteness of the information, and the aggregation or summarization of information from multiple sources.

The first source type is related to the reliability of information: uncertainty can be present in the factual knowledge (i.e., the set of assertions or facts) as a result of ill-defined concepts in the observations (fuzziness) or due to inaccuracy and poor reliability of the instruments used to make the observations (randomness). Randomness and fuzziness reflect different aspects of uncertainty. Randomness deals with the uncertainty of whether a given element belongs to a well-defined set (event). Fuzziness deals with the uncertainty derived from the partial membership of a given element to a set whose boundaries are not sharply defined. Uncertainty can also occur in the knowledge base (i.e., the rule set) as a result of weak implications when the expert or model builder is unable to establish a strong correlation between premise and conclusion. In most expert systems the degree of implication is artificially expressed as a scalar value on an interval (certainty factor, probability, etc.). This value represents the change from the strict implication for all  $x$ ,  $A(x) \rightarrow B(x)$ , to the weaker statement for most  $x$ , or usually, for all  $x$ ,  $A(x) \rightarrow B(x)$ . The latter statement is not categorical and allows the possibility of exceptions to the rule. Thus, the logical implication has now been changed into a plausible implication or disposition (2). A natural way to express such a degree of implication is achieved by using fuzzy quantifiers such as "most," "almost," etc. (3). A fuzzy quantifier is a fuzzy number representing the relative cardinality of the subset of elements in the universe of discourse that usually satisfy the given property, i.e., the implication. Uncertainty in the data can be compounded by aggregating uncertain data in the premise, by propagating certainty measures to the conclusion, and by consolidating the final certainty measure of conclusions derived from different rules. Triangular norms and conorms (4,5) can be used to generalize the conjunction and disjunction operators that provide the required aggregation capabilities.

A description of their characteristics is provided under Personal Perspective.

The second type of uncertainty is caused by the inherent imprecision of the rule-representation language. If rules are not expressed in a formal language, their meaning cannot be interpreted exactly. This problem has been partially addressed by the theory of approximate reasoning that, in light of imprecise fact and rule descriptions, allows one to make weaker inferences based on a generalized *modus ponens* (6).

The third type of uncertainty is caused by the incompleteness of the information. This type of uncertainty has generally been modeled by non-numerical characterizations, such as Doyle's reasoned assumptions (7).

The fourth type of uncertainty arises from the aggregation of information from different knowledge sources or experts. When unconditional statements (facts) are aggregated, three potential problems can occur: the closure of the representation may no longer be preserved when the facts to be aggregated have different granularity (the single-valued certainty measures of the facts may change into an interval-valued certainty measure of the aggregated fact); the aggregation of conflicting statements may generate a contradiction that should be detected; and the rule of evidence combination may create an overestimated certainty measure of the aggregated fact if a normalization is used to eliminate or hide a contradiction (8,9). The first two problems are typical of single-valued numerical approaches, whereas the last problem is found in the two-valued approach proposed by Dempster (10). All these approaches are discussed in the following section.

### State of the Art in Reasoning with Uncertainty

The existing approaches to represent uncertainty are divided into two basic categories: numerical characterizations and symbolic (i.e., nonnumerical) characterizations.

Among the numerical approaches, there are three distinct types: the one-valued, the two-valued, and the fuzzy-valued approach. Some of the more traditional techniques found among the one-valued approaches are Bayes's rule (11), the modified Bayesian rule (12), and confirmation theory (13). A more recent trend is exemplified by the two-valued approaches: Dempster-Shafer theory (10,14), evidential reasoning (15), consistency and plausibility (probability bounds) (16), and evidence space (17). The fuzzy-valued approaches include the necessity and possibility theory (18,19) and the linguistic variable approach (6). With numerical representations it is possible to define a calculus that provides the mechanism for propagating the uncertainty of the information through the reasoning process. Similarly, the use of aggregation operators provides a summary of the information, which can then be ranked against other such summaries to perform rational decisions. Such a numerical representation, however, cannot provide a clear explanation of the reasons that led to a given conclusion.

Models based on symbolic representations, on the other hand, are mostly designed to handle the aspect of uncertainty derived from the incompleteness of the information. They are generally inadequate to handle the case of imprecise information since they lack any measure to quantify confidence levels (7). Among the symbolic characterizations, there are two distinct approaches: the formal approach, such as reasoned assumptions (7), and default reasoning (20) (qv) and the heuristic approach, such as the theory of endorsements (21,22). The



formal approach has a corresponding logic theory that determines the mechanism by which inferences (theorems) can be proven or believed to be true. The heuristic approach has a set of context-dependent rules to define the way by which frame-like (see Frame theory) structures (endorsements) can be combined, added, or removed. The symbolic representations are more suitable for providing a trace from the sources of the information through the various inference paths to the final conclusions. However, no calculus can be defined for the propagation, aggregation, and ranking of such uncertain information. The only available partial solution is the use of context-dependent rules to determine how each piece of evidence can be compared or summarized.

In this entry selected approaches to the representation of uncertainty are illustrated and discussed.

**Bayes' Rule.** Given a set of hypotheses  $\mathbf{H} = \{h_1, h_2, \dots, h_n\}$  and a sequence of pieces of evidence  $\{e_1, e_2, \dots, e_m\}$ , Bayes's rule (see Bayesian decision theory), derived from the formula of conditional probability, states that the posterior probability  $P(h_i|e_1, e_2, \dots, e_m)$  can be derived as a function of the conditional probabilities  $P(e_1, e_2, \dots, e_m|h_i)$  and the prior probability  $P(h_i)$ :

$$P(h_i|e_1, e_2, \dots, e_m) = \frac{P(e_1, e_2, \dots, e_m|h_i)P(h_i)}{\sum_{i=1}^n P(e_1, e_2, \dots, e_m|h_i)P(h_i)} \quad (1)$$

The Bayesian approach is based on two fundamental assumptions shown below.

Each hypothesis  $h_i$  is mutually exclusive with any other hypothesis in the set  $\mathbf{H}$  and the set of hypotheses  $\mathbf{H}$  is exhaustive, i.e.,

$$P(h_i, h_j) = 0 \quad \text{for } i \neq j \quad (2)$$

$$\sum_{i=1}^n P(h_i) = 1 \quad (3)$$

Each piece of evidence  $e_j$  is conditionally independent under each hypothesis, i.e.,

$$P(e_1, e_2, \dots, e_m|h_i) = \prod_{j=1}^m P(e_j|h_i) \quad (4)$$

This method requires a large amount of data to determine the estimates for the prior and conditional probabilities. Such a requirement becomes manageable when the problem can be represented as a sparse Bayesian network that is formed by a hierarchy of small clusters of nodes. In this case the dependencies among variables (nodes in the network) are known and only the explicitly required conditional probabilities must be obtained (23). The Bayesian approach still exhibits all the same shortcomings discussed in the modified Bayesian approach.

**Modified Bayesian Rule.** In addition to Eqs. 2, 3, and 4 needed by the original Bayesian rule, the modified Bayesian approach used in PROSPECTOR (qv) also requires that each piece of evidence  $e_j$  be conditionally independent under the negation of each hypothesis, i.e.,

$$P(e_1, e_2, \dots, e_m|\neg h_i) = \prod_{j=1}^m P(e_j|\neg h_i) \quad (5)$$

The modified Bayesian approach is based on a variation of the odds-likelihood formulation of Bayes's rule. When all the pieces of evidence are certainly true, this formulation defines the posterior odds as

$$\begin{aligned} O(h_i|e_1, e_2, \dots, e_m) &= \frac{P(e_1|h_i)}{P(e_1|\neg h_i)} \frac{P(e_2|h_i)}{P(e_2|\neg h_i)} \dots \\ &\quad \frac{P(e_n|h_i)}{P(e_n|\neg h_i)} \frac{P(h_i)}{P(\neg h_i)} \quad (6) \\ &= \lambda_{1,i} \lambda_{2,i} \dots \lambda_{n,i} O(h_i) \end{aligned}$$

where

$\lambda_{j,i} = P(e_j|h_i)/P(e_j|\neg h_i)$  is the likelihood ratio of  $e_j$  for hypothesis  $h_i$

$O(h_i) = P(h_i)/P(\neg h_i)$  is the odds on hypothesis  $h_i$ .

An analogous odds-likelihood formulation is derived for the case when all the pieces of evidence are certainly false:

$$\begin{aligned} O(h_i|\neg e_1, \neg e_2, \dots, \neg e_m) &= \frac{P(\neg e_1|h_i)}{P(\neg e_1|\neg h_i)} \frac{P(\neg e_2|h_i)}{P(\neg e_2|\neg h_i)} \\ &\quad \dots \frac{P(\neg e_n|h_i)}{P(\neg e_n|\neg h_i)} \frac{P(h_i)}{P(\neg h_i)} \quad (7) \\ &= \lambda_{1,i}^* \lambda_{2,i}^* \dots \lambda_{n,i}^* O(h_i) \end{aligned}$$

The likelihood ratio  $\lambda_{j,i}$  measures the sufficiency of a piece of evidence  $e_j$  to prove hypothesis  $h_i$ . Similarly,  $\lambda_{j,i}^*$  measures the necessity of such a piece of evidence to prove the given hypothesis (12).

Formulas 6 and 7 assume that evidence  $e_j$  is precise [i.e.,  $P(e_j) = \{0, 1\}$ ]. This is not the case in most expert-system applications. Therefore, the above formulas must be modified to accommodate uncertain evidence. This is accomplished by using a linear interpolation formula. For the case of single evidence the posterior probability  $P(h_i|e_j)$  is computed as

$$P(h_i|e_j') = P(h_i|e_j)P(e_j|e_j') + P(h_i|\neg e_j)P(\neg e_j|e_j') \quad (8)$$

where  $P(e_j|e_j')$  is the user's assessment of the probability that the evidence  $e_j$  is true given the relevant observation  $e_j'$ .

An effective likelihood ratio,  $\lambda'_{j,i}$ , is calculated from the posterior odds:

$$\lambda'_{j,i} = \frac{O(h_i|e_j')}{O(h_i)} \quad (9)$$

The posterior odds for all the evidence is then computed as

$$O(h_i|e'_1, e'_2, \dots, e'_m) = O(h_i) \prod_{j=1}^m \lambda'_{j,i} \quad (10)$$

Formula 8, however, requires a modification because it over-constrains the input requested from the user. In fact, the user must specify

$O(h_i)$ , the prior odds on  $h_i$  from which  $P(h_i)$  can be derived;  
 $\lambda_{j,i}$ , the measure of sufficiency from which  $P(h_i|e_j)$  can be derived;

$\lambda_{j,i}^*$ , the measure of necessity from which  $P(h_i|\neg e_j)$  can be derived; and

$O(e_j)$ , the prior odds on  $e_j$  from which  $P(e_j)$  can be derived.

These requirements are equivalent to specifying a line in the coordinates  $[P(e|e'), P(h_i|e')]$  with three points:

$$(0, P(h_i|\neg e_j)), (P(e_j), P(h_i)), (1, P(h_i|e_j))$$

The modification adopted in this approach to prevent the user's inconsistencies is to change Eq. 8 into a piecewise linear function defined by two line segments passing through the above three points (12).

In an analysis of this approach, Pednault, Zucker, and Muresan (24) concluded that the assumptions of conditional independence of the evidence both under the hypotheses and under the negation of the hypotheses (as required by the Bayesian updating) were inconsistent with the other assumptions of an exhaustive and mutually exclusive space of hypotheses. Specifically, Pednault proved that, under these assumptions, if there were more than two competing hypotheses, no probabilistic update could take place, i.e.,

$$P(e_j|h_i) = P(e_j|\neg h_i) = P(e_j) \quad \text{for all } i, j$$

which implies:

$$\lambda_{j,i} = 1 \quad \text{for all } i, j$$

However, Glymour (25) obtained a pathological counterexample to Pednault's statement, finding a fault in the original proof of Hussain's theorem that constituted the basis for Pednault's results. Johnson (26) extended this analysis by first showing that there are also nonpathological counterexamples that refute Pednault's results. However, Johnson proved that under the same assumptions used in Pednault's work, for every hypothesis  $h_i$  there is at most one piece of evidence  $e_j$  that produces updating for  $h_i$ .

Pearl has argued (11) that Eq. 5, requiring the conditional independence of the evidence under the negation of the hypotheses, is overrestrictive. By discarding this assumption, Pearl has derived new, more promising results. However, the assumption of conditional independence of the evidence under the hypotheses was still required.

The Bayesian approach has various shortcomings. The assumptions on which it is based are not easily satisfiable, e.g., if the network contains multiple paths linking a given evidence to the same hypothesis, the independence Eqs. 4 and 5 are violated. Similarly, Eqs. 2 and 3, requiring the mutual exclusiveness and exhaustiveness of the hypotheses, are not very realistic; Eq. 2 would not hold if more than one hypothesis could occur simultaneously and is as restrictive as the single-fault assumption of the simplest diagnosing systems. Equation 2 implies that every possible hypothesis is a priori known, and it would be violated if the problem domain were not suitable to a closed-world assumption. Perhaps the most restrictive limitation of the Bayesian approach is its inability to represent ignorance (i.e., noncommitment as illustrated by its two-way betting interpretation (27)). The two-way betting interpretation of the Bayesian approach consists of regarding the assignment of probability  $p$  to event  $A$  as the willingness of a rational agent to accept any of the two following bets.

If you pay me \$ $p$ , then I agree to pay you \$1 if  $A$  is true.

If you pay me \$(1 -  $p$ ), then I agree to pay you \$1 if  $A$  is false.

The first bet represents the belief that the probability of  $A$  is not larger than  $p$ , the second bet represents the belief that the probability of  $A$  is not smaller than  $p$ .

Instead of being explicitly represented, ignorance is hidden in prior probabilities. Further shortcomings are represented by the fact that it is impossible to assign any probability to disjunctions, i.e., to nonsingletons, which implies the requirement for a uniform granularity of evidence. Finally, as was pointed out by Quinlan (16), in this approach conflictive information is not detected but simply propagated through the network.

**Confirmation Theory.** The certainty-factor (CF) approach (13), used in MYCIN (qv), is based on confirmation theory. The certainty factor  $CF(h, e)$  of a given hypothesis  $h$  is the difference between a measure of belief  $MB(h, e)$  representing the degree of support of a (favorable) evidence  $e$  and a measure of disbelief  $MD(h, e)$  representing the degree of refutation of an (unfavorable) evidence  $e$ .  $MB$  and  $MD$  are monotonically increasing functions that are respectively updated when the new evidence supports or refutes the hypothesis under consideration. The certainty factor  $CF(h, e)$  is defined as:

$$CF(h, e) = \begin{cases} 1 & \text{if } P(h) = 1 \\ MB(h, e) & \text{if } P(h|e) > P(h) \\ 0 & \text{if } P(h|e) = P(h) \\ -MD(h, e) & \text{if } P(h|e) < P(h) \\ -1 & \text{if } P(h) = 0 \end{cases} \quad (11)$$

The measures of belief  $MB$  and measure of disbelief  $MD$  could be interpreted as a relative distance on a bounded interval. Given an interval  $[A, B]$  and a reference point  $R$  within the interval, the relative distance  $d(X, R)$  between any arbitrary point  $X$  within the interval and the reference  $R$  can be defined as

$$d(X, R) = \begin{cases} (X - R)/(B - R) & \text{if } X > R \\ 0 & \text{if } X = R \\ (R - X)/(R - A) & \text{if } X < R \end{cases} \quad (12)$$

By making the substitutions

$$A = 0 \quad B = 1 \quad R = P(h) \quad X = P(h|e)$$

the definition of the measure of belief ( $MB$ ) and measure of disbelief ( $MD$ ) can be obtained:

$$MB(h, e) = \begin{cases} \frac{P(h|e) - P(h)}{1 - P(h)} & \text{if } P(h|e) > P(h) \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

$$MD(h, e) = \begin{cases} \frac{P(h) - P(h|e)}{P(h)} & \text{if } P(h|e) < P(h) \\ 0 & \text{otherwise} \end{cases} \quad (14)$$

The CF was originally interpreted as the relative increase or decrease of probabilities. In fact, from Eqs. 11, 13, and 14 it can be shown that

$$P(h|e) = P(h) + CF(h, e)[1 - P(h)] \quad \text{for } CF(h, e) \geq 0 \quad (15)$$

$$P(h|e) = P(h) - |CF(h, e)|P(h) \quad \text{for } CF(h, e) \leq 0 \quad (16)$$

Too often the CF paradigm has been incorrectly used in reasoning systems, interpreting the CFs as absolute rather than incremental probability values. The original interpretation of the CF as a probability ratio, however, can no longer be preserved after the CFs have been aggregated using the heuristic combining functions provided in MYCIN (13).

Ishizuka, Fu, and Yao (28) have shown that these combining functions were an approximation of the classical Bayesian updating procedure, in which a term had been neglected. In their analysis it was concluded that the assumption of mutual independence of evidence was required for the correct use of this approach. The original definition of certainty factor is asymmetric and prevents commutativity. Another source of concern in the use of CFs is caused by the normalization of MBs and MDs before their arithmetic difference is computed. This normalization hides the difference between the cardinality of the set of supporting evidence and that of the set of refuting evidence.

Buchanan and Shortliffe (29) have proposed a change to the definition of CF and its rules of combination:

$$CF(h, e) = \frac{MB(h, e) - MD(h, e)}{1 - \min(MB(h, e), MD(h, e))} \quad (17)$$

$$CF_{\text{combine}}(x, y) = \begin{cases} x + y - xy & \text{for } x > 0, y > 0 \\ x + y & \text{for } x < 0, y > 0 \\ 1 - \min(|x|, |y|) & \text{or } x > 0, y < 0 \\ -CF_{\text{combine}}(-x, -y) & \text{for } x < 0, y < 0 \end{cases} \quad (18)$$

where

$$CF(h, e_1) = x \quad \text{and} \quad CF(h, e_2) = y$$

This new definition avoids the problem of allowing a single piece of negative (positive) evidence to overwhelm several pieces of positive (negative) evidence. However, it has even less theoretical justification or interpretation than the original formulas.

Recently, Heckerman (30) has derived a new definition for the CF that does allow commutativity and has a consistent probabilistic interpretation. The new definition is

$$CF(h, e) = \frac{P(h|e) - P(h)}{P(h|e)[1 - P(h)] + P(h)[1 - P(h|e)]} \quad (19)$$

There are still numerous serious problems that characterize this approach: the semantics of the CF, i.e., the interpretation of the number (ratio of probability, combination of utility values and probability); the assumptions of independence of the evidence; and the inability to distinguish between ignorance and conflict, both of which are represented by a zero CF.

This type of representation of uncertainty has also been advocated by Rich (31) as an alternative to default reasoning. In her work Rich claims that default reasoning could actually be better interpreted as likelihood reasoning, providing a uniform representation for statistical, prototypical, and definitional facts.

**Belief Theory.** The belief theory (see Belief systems) proposed by Shafer (14) was developed within the framework of Dempster's work on upper and lower probabilities induced by a multivalued mapping. The one-to-many nature of the mapping is the fundamental reason for the inability of applying the well-known theorem of probability that determines the probability density of the image of one-to-one mappings. In fact, given a differentiable strictly increasing or strictly decreasing function  $\phi$  on an interval  $I$  and a continuous random variable  $X$  with a density  $f$  such that  $f(x) = 0$  for any  $x$  outside  $I$ , the density function  $g$  can be computed as

$$g(y) = f(x) \left| \frac{dx}{dy} \right|, \quad y \in \phi(I) \text{ and } x = \phi^{-1}(y)$$

In this context the lower probabilities have been identified as epistemic probabilities and associated with a degree of belief. This formalism defines certainty as a function that maps subsets of a space of propositions  $\Theta$  and on the  $[0, 1]$  scale. The sets of partial beliefs are represented by mass distributions of a unit of belief across the propositions in  $\Theta$ . This distribution is called basic probability assignment (bpa). The total certainty over the space is 1. A nonzero bpa can be given to the entire space  $\Theta$  to represent the degree of ignorance. Given a space of propositions  $\Theta$ , referred to as frame of discernment, a function  $m: 2^\Theta \rightarrow [0, 1]$  is called a bpa if

$$m(\emptyset) = 0 \quad \text{where } \emptyset \text{ is the empty set}$$

$$0 < m(A) < 1 \quad (20)$$

$$\sum_{A \subseteq \Theta} m(A) = 1$$

The certainty of any proposition  $B$  is then represented by the interval  $[\text{Bel}(B), P^*(B)]$ , where  $\text{Bel}(B)$  and  $P^*(B)$  are defined as

$$\text{Bel}(B) = \sum_{x \subseteq B} m(x) \quad P^*(B) = \sum_{x \cap B \neq \emptyset} m(x) \quad (21)$$

From the above definitions the following relation can be derived:

$$\text{Bel}(B) = 1 - P^*(\neg B) \quad (22)$$

If  $m_1$  and  $m_2$  are two bpas induced from two independent sources, a third bpa,  $m(C)$ , expressing the pooling of the evidence from the two sources, can be computed by using Dempster's rule of combination (10):

$$m(C) = \frac{\sum_{A_i \cap B_j = C} m_1(A_i) m_2(B_j)}{1 - \sum_{A_i \cap B_j = \emptyset} m_1(A_i) m_2(B_j)} \quad (23)$$

Dempster's rule of combination normalizes the intersection of the bodies of evidence from the two sources by the amount of nonconflicting evidence between the sources. This amount is represented by the denominator of the formula.

There are two problems with the belief-theory approach. The first problem stems from computational complexity: In the general case, the evaluation of the degree of belief and upper probability requires time exponential in  $|\Theta|$ , the cardinality of the hypothesis set (frame of discernment). This is caused by the need of (possibly) enumerating all the subset and superset of a given set. Barnett (32) showed that when the frame of discernment is discrete, the computational-time complexity could be reduced from exponential to linear by combining the belief functions in a simplifying order. Strat (33) proved that the complexity could be reduced to  $O(n^2)$ , where  $n$  is the number of atomic propositions, i.e., intervals of unit length, when the frame of discernment is continuous. In both cases, however, these results were achieved by introducing various assumptions about the type and structure of the evidence to be combined and about the hypotheses to be supported. As a result, in addition to the requirements of mutually exclusive hypotheses and independent evidence needed by this approach, the following constraints must be included: for the case of discrete frame of discernment, each piece of evidence is assumed to support only a singleton proposition or its negation rather than disjunctions of propositions (i.e., propositions with larger granularity), and for the case of continuous frame of discernment, only contiguous intervals along the number line

can be included in the frame of discernment and thus receive support from the evidence. The second problem in this approach results from the normalization process present in both Dempster's and Shafer's works. Zadeh (8,9) has argued that this normalization process can lead to incorrect and counterintuitive results. By removing the conflictive parts of the evidence and normalizing the remaining parts, important information is discarded rather than being dealt with adequately. A proposed solution to this problem is to avoid completely the normalization process by maintaining an explicit measure of the amount of conflict and by allowing the remaining information to be subnormal [i.e.,  $\text{Bel}(\Theta) < 1$ ]. Zadeh (9) has proposed a test to determine the conditions of applicability of Dempster's rule of combination. Dubois and Prade (34) have also shown that the normalization process in the rule of evidence combination creates a sensitivity problem, where assigning a zero value or a very small value to a bpa causes very different results.

Ginsberg (35) has proposed the use of the Dempster-Shafer approach as an alternative to nonmonotonic logic (see Reasoning, nonmonotonic). This suggestion is an extension to Rich's idea of interpreting default reasoning as likelihood reasoning (31). In his work Ginsberg provides a rule for propagating the lower and upper bounds through a reasoning chain or graph. His result is based on the interpretation of a production rule as a conditional probability rather than as a material implication. Smets (36) has further explained the relations between belief functions, plausibilities, necessities, and possibilities and has extended Dempster's concepts to handle the case when the evidence is a fuzzy set (37).

**Evidential Reasoning.** Evidential reasoning, proposed by Garvey, Lowrance, and Fischler (15,38) adopts the evidential interpretation of the degrees of belief and upper probabilities. Fundamentally based on the Dempster-Shafer theory, this approach defines the likelihood of a proposition  $A$  as a subinterval of the unit interval  $[0, 1]$ . The lower bound of this interval is the degree of support of the proposition,  $S(A)$ , and the upper bound is its degree of plausibility,  $\text{Pl}(A)$ . The likelihood of a proposition  $A$  is written as  $A_{[S(A), \text{Pl}(A)]}$ . The following sample of interval-valued likelihoods illustrates the interpretation provided by this approach:

$A_{[0,1]}$	No knowledge at all about $A$
$A_{[0,0]}$	$A$ is false
$A_{[1,1]}$	$A$ is true
$A_{[0.3,1]}$	The evidence partially supports $A$
$A_{[0,0.7]}$	The evidence partially supports $\neg A$
$A_{[0.3,0.7]}$	The evidence simultaneously provides partial support for $A$ and $\neg A$
$A_{[0.3,0.3]}$	The probability of $A$ is exactly 0.3

Given two statements  $A_{[S(A), \text{Pl}(A)]}$  and  $B_{[S(B), \text{Pl}(B)]}$ , the set of inference rules corresponding to the logical operations on these statements are defined (15) as:

Intersection	$\text{AND}(A, B)_{[\max(0, S(A)+S(B)-1), \min(\text{Pl}(A), \text{Pl}(B))]}$
Union	$\text{OR}(A, B)_{[\max(S(A), S(B)), \min(1, \text{Pl}(A)+\text{Pl}(B))]}$
Negation	$\text{NOT}(A)_{[1-\text{Pl}(A), 1-S(A)]}$

This approach is based on the Dempster-Shafer (DS) theory. When distinct bodies of evidence must be pooled, this approach uses the same DS techniques, requiring the same normalization process that was criticized by Zadeh.

**Evidence Space.** Evidence space, proposed by Rollinger (17), represents the uncertainty of a statement as a point in a two-dimensional space. The  $(X, Y)$  coordinates of this space represent the positive or supporting evidence ( $E+$ ) and the negative or disconfirming evidence ( $E-$ ) available for any given proposition, respectively. The evidence space is a  $[0, 1] \times [0, 1]$  square whose four vertices represent ignorance  $(0, 0)$ , absolute certainty in the support  $(1, 0)$ , absolute certainty in the refutation  $(0, 1)$ , and maximum conflictive evidence  $(1, 1)$ . The diagonal line defined by the equation  $x + y - 1 = 0$  represents the locus of probability points, the sum of whose coordinates is 1.

If the dimensions of the evidence space ( $E+$ ,  $E-$ ) represent the necessary evidence, i.e., the lower bounds of the degree of support and refutation ( $S(E)$ ,  $S(\neg E)$ ), the evidence space is reduced to the lower left triangle. Its three vertices  $(0, 0)$ ,  $(1, 0)$ , and  $(0, 1)$  represent ignorance, absolute support, and absolute refutation, respectively. The maximum amount of conflict is given by the point  $(0.5, 0.5)$ . On the other hand, if the dimensions of the evidence space ( $E+$ ,  $E-$ ) represent the possible evidence, i.e., the upper bounds of the degree of support and refutation ( $\text{Pl}(E)$ ,  $\text{Pl}(\neg E)$ ), the evidence space is reduced to the upper-right triangle. Its three vertices  $(1, 1)$ ,  $(1, 0)$ , and  $(0, 1)$  represent ignorance, absolute support, and absolute refutation, respectively. The maximum amount of conflict is again given by the point  $(0.5, 0.5)$ . If the lower bounds are equated to the upper bounds, i.e.,  $(S(E), S(\neg E)) = (\text{Pl}(E), \text{Pl}(\neg E))$ , a new set of coordinates  $(P(E), P(\neg E))$ , representing Bayesian probability, can be obtained. In this new set of coordinates the evidence space collapses to the diagonal line  $x + y - 1 = 0$  that is the intersection of the two triangles and that indeed represents the probability line.

Rollinger (17) suggests the use of a distance to verify the validity of any given premise in a rule. This approach, however, does not suggest any way of aggregating evidence, propagating uncertainty through an inference chain, selecting an appropriate metric of similarity between patterns and data, etc.

**Necessity and Possibility Theory.** Necessity and possibility, proposed by Zadeh (36,37), measure the degree of entailment and intersection of two fuzzy propositions represented by their normalized possibility distributions. Normal necessity and possibilities correspond to consonant belief and plausibility functions, respectively. Given two fuzzy propositions  $A$  and  $B \subset U$ , characterized by their possibility distributions  $\mu_A(u_1)$  and  $\mu_B(u_2)$ , their degree of matching is represented by the interval  $[\text{Nec}(A|B), \text{Poss}(A|B)]$  (38), where

$$\begin{aligned} \text{Nec}(A|B) &= \infimum(B \rightarrow A) \\ &= \infimum_{u_1=u_2} \max(1 - \mu_B(u_2), \theta_A(u_1)) \quad (24) \end{aligned}$$

$$\begin{aligned} \text{Poss}(A|B) &= \supremum(B \cap A) \\ &= \supremum_{u_1=u_2} \min(\mu_B(u_2), \mu_A(u_1)) \quad (25) \end{aligned}$$

From the above definition, it is possible to derive for necessity and possibility the same duality observed between belief functions and upper probabilities (see formula 22), i.e.,

$$\text{Nec}(A|B) = 1 - \text{Poss}(\neg A|B) \quad (26)$$

The intersection-of-necessity measures and the union-of-possibility measures provide tighter bounds than those obtained by the intersection of belief functions and the union-of-plausibility functions (39).

**Reasoned Assumptions.** Among the non-numerical representations of uncertainty, two approaches typify the characterization of uncertain information in a purely symbolic manner: reasoned assumptions and theory of endorsements.

In the reasoned-assumption approach proposed by Doyle (7), the uncertainty embedded in an implication is (partially) removed by listing all the exceptions to that rule. When this is not possible, assumptions are used to show typicality of a value (default values) and defeasibility of a rule (liability to defeat of a reason). When an assumption used in the deductive process is found to be false, nonmonotonic mechanisms are used to keep the integrity of the database of statements. Assumption-based systems can cope with the case of incomplete information, but they are inadequate to handle the case of imprecise information. In particular, they cannot integrate probabilistic information with reasoned assumptions. Furthermore, these systems rely on the precision of the defaulted value. On the other hand, when specific information is missing, the system should be able to use analogous or relevant information inherited from some higher level concept. This surrogate for the missing information is generally fuzzy or imprecise and only provides some elastic constraints on the value of the missing information. It is recognized by Doyle (7) that assumption-based systems lack facilities for computing degrees of belief, which "may be necessary for summarizing the structure of large sets of admissible extensions as well as for quantifying confidence levels."

**Theory of Endorsements.** A different approach to uncertainty representation was recently proposed by Cohen and is based on a purely qualitative theory of endorsements (21,22). Endorsements are based on the explicit recording of the justifications for a statement, as in a truth-maintenance system (TMS). In addition, endorsements classify the justification according to the type of evidence (for and against a proposition), the possible actions required to solve the uncertainty of that evidence, and other related features. Endorsements provide a good mechanism for explanations, since they create and maintain the entire history of justifications (reasons for believing or disbelieving a proposition) and the relevance of any proposition with respect to a given goal. Endorsements are divided into five classes: rules, data, task, conclusion, and resolution endorsements. However, combination of endorsements in a premise, propagation of endorsements to a conclusion, and ranking of endorsements must be explicitly specified for each particular context, creating potential combinatorial problems.

### Desiderata for Reasoning with Uncertainty

From the previous review of the state of the art it is possible to derive desiderata (i.e., the list of requirements to be satisfied by the ideal formalism for representing uncertainty and making inference with uncertainty). Some of the existing approaches to reasoning with uncertainty will be compared against the list of requirements.

1. The combination rules should not be based on global assumptions of evidence independence.
2. The combination rules should not assume the exhaustiveness and exclusiveness of the hypotheses.
3. There should be an explicit representation of the amount of evidence for supporting and for refuting any given hypothesis.

4. There should be an explicit representation of the reasons for supporting and for refuting any given hypothesis.
5. The representation should allow the user to describe the uncertainty of information at the available level of detail (i.e., allowing heterogeneous information granularity).
6. There should be an explicit representation of consistency. Some measure of consistency or compatibility should be available to detect trends of potential conflicts and to identify essential contributing factors in the conflict.
7. There should be an explicit representation of ignorance to allow the user to make noncommitting statements, i.e., to express the user's lack of conviction about the certainty of any of the available choices or events. Some measure of ignorance, similar to the concept of entropy, should be available to guide the gathering of discriminant information.
8. There should be a clear distinction between a conflict in the information (i.e., violation of consistency) and ignorance about the information.
9. There should be a second-order measure of uncertainty. It is important to measure the uncertainty of the information as well as the uncertainty of the measure itself.
10. The representation must be, or at least must appear to be, natural to the user to enable him/her to describe uncertain input and to interpret uncertain output. The representation must also be natural to the expert to enable consistent weights or reasons for the uncertainty to be elicited. The semantics for any numerical operation used to propagate and summarize uncertainty should be clear.
11. The syntax and semantics of the representation of uncertainty should be closed under the rules of combinations.
12. Making pairwise comparisons of uncertainty should be feasible since the induced ordinal or cardinal ranking is needed for performing any kind of decision-making activities.
13. The traceability of the aggregation and propagation of uncertainty through the reasoning process must be available to resolve conflicts or contradictions, to explain the support of conclusions, and to perform metareasoning for control.

Table 1 summarizes the comparison among seven of the numerical and two of the symbolic approaches mentioned in this entry.

### Personal Perspective on Reasoning with Uncertainty

In building expert-system architectures three distinct layers must be defined: representation, inference, and control layers. The treatment of uncertainty in expert systems must address each of these layers. The majority of the approaches discussed above do not properly cover these issues. Some approaches lack expressiveness in their representation paradigm. Other approaches require unrealistic assumptions to provide uniform combination rules defining the plausible inferences. Specifically, the non-numerical approaches are inadequate to represent and summarize measures of uncertainty. The numerical approaches generally tend to impose some restrictions on the type and structure of the information (e.g., mutual exclusiveness of hypotheses, conditional independence of evidence). Most numerical approaches represent uncertainty as a

Table 1. Comparison of Selected Uncertainty Representations

Uncertainty Representation	Requirement Number												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Modified Bayesian	N	N	N	N	N	N	N	N	N	Y	Y	Y	N
Confirmation	N	Y	Y/N <sup>a</sup>	N	N	N	Y	N	N	N	N	Y	N
Upper and lower probabilities	N	N	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N
Evidential reasoning	N	N	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N
Probability bounds	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N
Fuzzy necessity and possibility	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	N
Evidence space	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	N
Reasoned assumptions	Y	Y	N	Y	N	Y	N	N	N	Y	Y	N	Y
Endorsements	Y	Y	N	Y	N	Y	N	N	N	Y	Y	N	Y

<sup>a</sup> The Certainty Factor used in Confirmation Theory was initially obtained from the difference of the MB and the MD. These two measures provided an explicit representation of support and refutation. However, more recently, the certainty factor was directly obtained from the user, and new certainty factors were calculated from the old ones, without maintaining the two measures (MB and MD). This is the reason for having the value Y/N in this entry.

precise quantity (scalar or interval) on a given scale. They require the user or expert to provide a precise yet consistent numerical assessment of the uncertainty of the atomic data and of their relations. The output produced by these systems is the result of laborious computations guided by well-defined calculi and appears to be equally precise. However, given the difficulty in consistently eliciting such numerical values from the user, it is clear that these models of uncertainty require an unrealistic level of precision that does not actually represent a real assessment of the uncertainty. An alternative solution to this problem is now illustrated.

**Representing Uncertainty Granularity.** Bonissone (4) has presented a formalism to represent any truth-functional calculus of uncertainty in terms of a selection of a negation operator and two elements from parameterized families of triangular norms and conorms. These functions take as arguments real-number values on the  $[0, 1]$  interval, which must be initially provided by the user or the expert. However, to avoid the fake-precision assumption required by most existing numerical approaches, the use of various term sets has been proposed. Each term set defines a different verbal scale of certainty by providing a different set of linguistic estimates of the likelihood of any given statement. Thus, the selection of a term set determines the uncertainty granularity (i.e., the finest level of distinction among different quantifications of uncertainty). The semantics for the elements of each term set are given by fuzzy numbers on the  $[0, 1]$  interval. The values of the fuzzy numbers have been determined from the results of a psychological experiment aimed at the consistent use of linguistic probabilities (40). The applicability of triangular norms and conorms is extended to fuzzy numbers by using a parametric representation for fuzzy numbers that allows closed-form solutions for arithmetic operation.

**Aggregation Operators: Conjunctions and Disjunctions.** The generalizations of conjunctions and disjunctions play a vital role in the management of uncertainty in expert systems; they are used in evaluating the satisfaction of premises, in propagating uncertainty through rule chaining, and in consolidating the same conclusion derived from different rules. More specifically, they provide the answers to the following questions.

When the premise is composed of multiple clauses, how to aggregate the degree of certainty  $x_i$  of the facts matching

the clauses of the premise? (That is, what is the function  $T(x_1, \dots, x_n)$  that determines  $x_p$ , the degree of certainty of the premise?).

When a rule does not represent a logical implication, but rather an empirical association between premise and conclusion, how to aggregate the degree of satisfaction of the premise  $x_p$  with the strength of the association  $s_r$ ? (That is, what is the function  $G(x_p, s_r)$  that propagates the uncertainty through the rule?).

When the same conclusion is established by multiple rules with various degrees of certainty  $y_1, \dots, y_m$ , how to aggregate these contributions into a final degree of certainty? (That is, what is the function  $S(y_1, \dots, y_m)$  that consolidates the certainty of that conclusion?).

Triangular norms (T-norms) and triangular conorms (T-conorms) are the most general families of binary functions that satisfy the requirements of the conjunction and disjunction operators, respectively. T-norms and T-conorms are two-place functions from  $[0, 1] \times [0, 1]$  to  $[0, 1]$  that are monotonic, commutative, and associative. Their corresponding boundary conditions satisfy the truth tables of the logical AND and OR operators.

**Conjunction and Propagation Using Triangular Norms.** The function  $T(a, b)$  aggregates the degree of certainty of two clauses in the same premise. This function performs an intersection operation and satisfies the conditions of a triangular norm (T-norm):

$$\begin{aligned}
 T(0, 0) &= 0 && \text{(boundary)} \\
 T(a, 1) &= T(1, a) = a && \text{(boundary)} \\
 T(a, b) &\leq T(c, d) \quad \text{if } a \leq c \text{ and } b \leq d && \text{(monotonicity)} \\
 T(a, b) &= T(b, a) && \text{(commutativity)} \\
 T(a, T(b, c)) &= T(T(a, b), c) && \text{(associativity)}
 \end{aligned}$$

Although defined as two-place functions, the T-norms can be used to represent the intersection of a larger number of clauses in a premise. Because of the associativity of the T-norms, it is possible to define recursively  $T(x_1, \dots, x_n, x_{n+1})$ , for  $x_1, \dots, x_{n+1} \in [0, 1]$ , as

$$T(x_1, \dots, x_n, x_{n+1}) = T(T(x_1, \dots, x_n), x_{n+1})$$

A special case of the conjunction is the detachment function  $G(x_p, s_r)$ , which attaches a certainty measure to the conclusion of a rule. This measure represents the aggregation of the certainty value of the premise of the rule  $x_p$  (indicating the degree



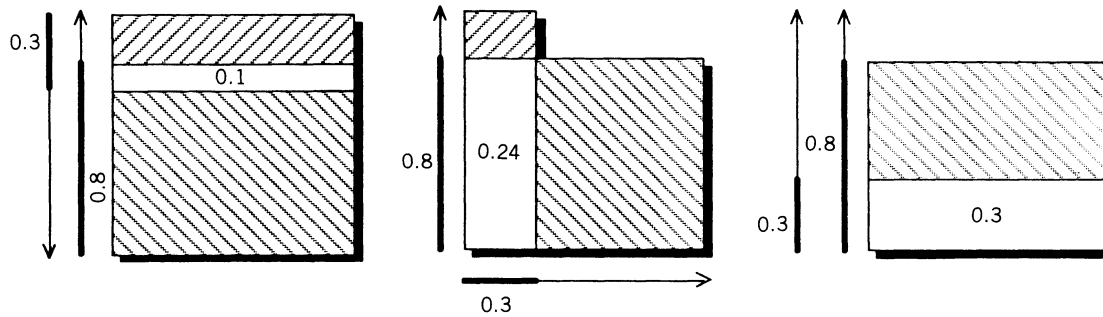


Figure 1. Geometric interpretation of  $T_1(0.3, 0.8)$ ,  $T_2(0.3, 0.8)$ , and  $T_3(0.3, 0.8)$ .

of fulfillment of the premise) with the strength of the rule  $s_r$  (indicating the degree of causal implication or empirical association of the rule). This function satisfies the same conditions of the T-norm (although it does not need to be commutative).

**Disjunction Using Triangular Conorms.** The function  $S(a, b)$  aggregates the degree of certainty of the (same) conclusions derived from two rules. This function performs a union operation and satisfies the conditions of a triangular conorm (T-conorm):

$$\begin{aligned} S(1, 1) &= 1 && \text{(boundary)} \\ S(0, a) &= S(a, 0) = a && \text{(boundary)} \\ S(a, b) &\leq S(c, d) \quad \text{if } a \leq c \text{ and } b \leq d && \text{(monotonicity)} \\ S(a, b) &= S(b, a) && \text{(commutativity)} \\ S(a, S(b, c)) &= S(S(a, b), c) && \text{(associativity)} \end{aligned}$$

A T-conorm can be extended to operate on more than two arguments in a manner similar to the extension for the T-norms. By using a recursive definition based on the associativity of the T-conorms, it is possible to define

$$S(y_1, \dots, y_m, y_{m+1}) = S(S(y_1, \dots, y_m), y_{m+1})$$

**Relationships between T-norms and T-conorms.** For suitable negation operations  $N(x)$ , such as  $N(x) = 1 - x$ , T-norms  $T$  and T-conorms  $S$  are duals in the sense of the following generalization of DeMorgan's (41) law:

$$S(a, b) = N[T(N(a), N(b))] \quad T(a, b) = N[S(N(a), N(b))]$$

This duality implies that the extensions of the intersection and union operators cannot be independently defined and they should, therefore, be analyzed as DeMorgan triples  $(T(\cdot, \cdot), S(\cdot, \cdot), N(\cdot))$  or, for a common negation operator like  $N(a) = 1 - a$ , as DeMorgan pairs  $(T(\cdot, \cdot), S(\cdot, \cdot))$ .

**Selecting Uncertainty Calculi.** Because of the difficulties in eliciting precise yet consistent numerical values from the user or expert, the use of term sets has been proposed. Each term set determines the finest level of specificity (i.e., the granularity) of the measure of certainty that the user/expert can consistently provide. This granularity limits the ability to differentiate between two similar calculi. Therefore, only a small finite subset of the infinite number of calculi produces notably different results. The number of calculi to be considered is a function of the uncertainty granularity.

This result has been confirmed by an experiment (4) where 11 different calculi of uncertainty, represented by their corresponding T-norms, were used with three term sets containing 5, 9, and 13 elements, respectively. For each of the three term sets, the T-norms were evaluated on the cross product of the

term-set elements, generating the closure of each T-norm. Each closure was compared with the closure of the adjacent T-norm and the number of differences were computed. The T-norms that did not exhibit significant differences were considered similar enough to be equivalent for any practical purpose. A threshold value determined the maximum percentage of differences allowed among members of the same equivalence class. Only three calculi generated sufficiently distinct results for those term sets that contained no more than nine elements.

The three calculi were defined by the following operators:

$$(T_1(a, b), S_1(a, b), N(a)) \quad (T_2(a, b), S_2(a, b), N(a)) \quad (T_3(a, b), S_3(a, b), N(a))$$

where  $N(a)$  is the negation operator  $N(a) = 1 - a$ , and the DeMorgan pairs of triangular norms  $T_i(a, b)$  and triangular conorms  $S_i(a, b)$  are

$$\begin{aligned} T_1(a, b) &= \max(0, a + b - 1) & S_1(a, b) &= \min(1, a + b) \\ T_2(a, b) &= ab & S_2(a, b) &= a + b - ab \\ T_3(a, b) &= \min(a, b) & S_3(a, b) &= \max(a, b) \end{aligned}$$

Since triangular norms and conorms are duals in the DeMorgan sense, they cannot be independently specified. Thus, to understand the meaning of each calculus, it is enough to analyze just one operator. A first interpretation suggests that  $T_1$  is appropriate to perform the intersection of lower probability bounds (10) or degrees of necessity (19). Similarly,  $T_3$  is appropriate to represent the intersection of upper probability bounds, or degrees of possibility. Here  $T_2$  is the classical probabilistic operator that assumes independence of the arguments; its dual T-conorm,  $S_2$ , is the usual additive measure for the union.

Figure 1 provides a geometric description of the meaning of the three T-norms. It illustrates the result of  $T_1(0.3, 0.8)$ ,  $T_2(0.3, 0.8)$ , and  $T_3(0.3, 0.8)$ . Here  $T_1$  captures the notion of worst case, where the two arguments are considered as mutually exclusive as possible (the dimensions on which they are measured are  $180^\circ$  apart);  $T_2$  captures the notion of independence of the arguments (their dimensions are  $90^\circ$  apart); and  $T_3$  captures the notion of best case, where one argument attempts to subsume the other one (their dimensions are collinear, i.e.,  $0^\circ$  apart).

## Conclusions

The sources of uncertainty have been analyzed and classified into four different types: the reliability of the information, the imprecision of the language used to represent it, the incompleteness of the information, and its aggregation. Non-numerical approaches were discussed in their roles of coping with incomplete information. A review of the state of the art in

representing the other three types of uncertainty provided a historic perspective in which past and current numerical approaches were described. The characteristics of these approaches were evaluated against desiderata.

It has also been suggested that the treatment of uncertainty in expert systems should address the same three layers of representation, inference, and control that typify expert-system architectures. The characterization of uncertainty measures as linguistic variables with fuzzy-valued semantics and the use of a given uncertainty calculus have addressed the representation and inference layers, respectively. The selection of the most appropriate calculus to be used must be addressed by the control layer.

However, in most expert systems the control layer has been procedurally embedded in the inference engine, thus preventing any opportunistic and dynamic change in ordering inferences and in aggregating uncertainty. Usually, the same type of aggregation operators (i.e., the same uncertainty calculus) is selected a priori and is used uniformly for any inference made by the expert system. The most recent trend in building expert systems is moving toward having a declarative representation for the control layer.

As an integral part of this layer, it is suggested to define a set of context-dependent rules that will select the most appropriate calculus for any given situation. Such a rule set will be relatively small since it must describe only the selection policies for a small number of calculi. The reduced number of calculi is the result of the analyzed trade-off between precision and complexity (4). These rules will rely on contextual information—such as the nature, reliability, and characteristics of the evidence sources—as well as on the meanings of the calculi that will be used in the inference layer.

## BIBLIOGRAPHY

1. P. P. Bonissone and R. M. Tong, "Editorial: Reasoning with uncertainty in expert systems," *Int. J. Man-Mach. Stud.* **22**(3), 241–250 (March 1985).
2. L. A. Zadeh, Syllogistic Reasoning in Fuzzy Logic and its Application to Reasoning with Dispositions, Electronics Research Laboratory Memorandum No. UCB/ERL/M84/17, University of California, Berkeley, 1984.
3. L. A. Zadeh, "A computational approach to fuzzy quantifiers in natural languages," *Comput. Math. Appl.* **9**(1), 149–184 (1983).
4. P. P. Bonissone and K. S. Decker, Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off Precision and Complexity, *Proceedings of the Workshop on Uncertainty and Probability in Artificial Intelligence*, pp. 57–66, University of California, Los Angeles, August 14–16, 1985, also in J. Lemmer and L. Kanal (eds.), *Uncertainty in Artificial Intelligence*, North-Holland, New York, 1986, pp. 217–247.
5. D. Dubois and H. Prade, "Criteria aggregation and ranking of alternatives in the framework of fuzzy set theory," *TIMS/Stud. Manag. Sci.* **20**, 209–240 (1984).
6. L. A. Zadeh, "Fuzzy logic and approximate reasoning (in memory of Grigor Moisil)," *Synthese* **30**, 407–428 (1975).
7. J. Doyle, "Methodological simplicity in expert system construction: The case of judgments and reasoned assumptions," *AI Mag.* **4**(2), 39–43, (Summer 1983).
8. L. A. Zadeh, "Review of books: A mathematical theory of evidence," *AI Mag.* **5**(3), 81–83 (1984).
9. L. A. Zadeh, A Simple View of the Dempster-Shafer Theory of Evidence and Its Implications for the Rule of Combinations, Berkeley Cognitive Science Report No. 33, Institute of Cognitive Science, University of California, Berkeley, November 1985.
10. A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *Ann. Math. Stat.* **38**(2), 325–339 (1967).
11. J. Pearl, Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach, *Proceedings of the Second National Conference on Artificial Intelligence*, Pittsburgh, PA, pp. 133–136, 1982.
12. R. O. Duda, P. E. Hart, and N. J. Nilsson, Subjective Bayesian Methods for Rule-Based Inference Systems, *AFIPS Conference Proceedings*, New York, June 1976, pp. 1075–1082.
13. E. H. Shortliffe and B. G. Buchanan, "A model of inexact reasoning in medicine," *Math. Biosci.* **23**, 351–379 (1975).
14. G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ, 1976.
15. T. D. Garvey, J. D. Lowrance, and M. A. Fischler, An Inference Technique for Integrating Knowledge from Disparate Sources, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., 1981, pp. 319–325.
16. J. R. Quinlan, Consistency and Plausible Reasoning, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, 1983, pp. 137–144.
17. C. R. Rollinger, How to Represent Evidence—Aspects of Uncertainty Reasoning, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, 1983, pp. 358–361.
18. L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets Sys.* **1**, 3–28 (1978).
19. L. A. Zadeh, Fuzzy Sets and Information Granularity, in M. M. Gupta, R. K. Ragade, and R. R. Yager (eds.), *Advances in Fuzzy Set Theory and Applications*, North-Holland, New York, 1979, pp. 3–18.
20. R. Reiter, "A logic for default reasoning," *Artif. Intell.* **13**, 81–132 (1980).
21. P. R. Cohen and M. R. Grinberg, "A theory of heuristics reasoning about uncertainty," *AI Mag.* **4**(2), 17–23 (Summer 1983).
22. P. R. Cohen and M. R. Grinberg, A Framework for Heuristics Reasoning about Uncertainty, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, 1983, pp. 355–357.
23. J. Pearl, How to Do with Probabilities What People Say You Can't, *The Second Conference on Artificial Intelligence Applications*, IEEE Computer Society, Miami Beach, FL, December 1985, pp. 1–12.
24. E. P. D. Pednault, S. W. Zucker, and L. V. Muresan, "On the independence assumption underlying subjective Bayesian updating," *Artif. Intell.* **16**, 213–222 (1981).
25. C. Glymour, "Independence assumptions and Bayesian updating," *Artif. Intell.* **25**, 95–99 (1985).
26. R. W. Johnson, Independence and Bayesian Updating Methods, *Proceedings of the Workshop on Uncertainty and Probability in Artificial Intelligence*, University of California, Los Angeles, August 14–16, 1985, pp. 28–30.
27. R. Giles, "Semantics for fuzzy reasoning," *Int. J. Man-Mach. Stud.* **17**(4), 401–415 (1982).
28. M. Ishizuka, K. S. Fu, and J. T. P. Yao, Inexact Inference for Rule-Based Damage Assessment of Existing Structure, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., 1981, pp. 837–842.
29. B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems*, Addison-Wesley, Reading, MA, 1984.
30. D. Heckerman, Probabilistic Interpretations for MYCIN's Certainty Factors, *Proceedings of the Workshop on Uncertainty and Probability in Artificial Intelligence*, University of California, Los Angeles, August 14–16, 1985, pp. 9–20.

31. E. Rich, Default Reasoning as Likelihood Reasoning, *Proceedings of the Third National Conference on Artificial Intelligence*, Washington, DC, August 22–26, 1983, pp. 348–351.
32. J. A. Barnett, Computational Methods for a Mathematical Theory of Evidence, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., 1981, pp. 868–875.
33. T. M. Strat, Continuous Belief Functions for Evidential Reasoning, *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, August 6–10, 1984, pp. 308–313.
34. D. Dubois and H. Prade, Combination and Propagation of Uncertainty with Belief Functions—A Reexamination, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, 1985, pp. 111–113.
35. M. L. Ginsberg, Non-Monotonic Reasoning Using Dempster's Rule, *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, August 6–10, 1984, pp. 126–129.
36. P. Smets, "The degree of belief in a fuzzy set," *Inf. Sci.* **25**, 1–19 (1981).
37. L. A. Zadeh, "Fuzzy sets," *Inf. Cntrl.* **8**, 338–353 (1965).
38. J. D. Lowrance and T. D. Garvey, Evidential Reasoning: A Developing Concept, *Proceedings of the International Conference on Cybernetics and Society*, Seattle, WA, 1982, pp. 6–9.
39. H. Prade, A Computational Approach to Approximate Reasoning and Plausible Reasoning with Applications to Expert Systems, *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-7**(3), 260–283 (May 1985).
40. R. Beyth-Marom, "How probable is probable? A numerical taxonomy translation of verbal probability expressions," *J. Forecast.* **1**, 257–269 (1982).
41. N. Jacobson, *Basic Algebra I*, W. H. Freeman, San Francisco 1974, p. 4.

P. BONISSONE  
General Electric

## REASONING, SPATIAL

Human beings spend much of their time solving spatial problems, such as finding their way around. Furthermore, they often seem to use spatial methods for solving problems analogically, as when they reason about graphs by drawing or imagining pictures. Unfortunately, little is known about this ability. It is not even known whether there is one ability here or a set of unrelated abilities. Consequently, this survey must be of fragmentary, preliminary work.

Tentatively this research area can be divided into these subheadings:

- visual object recognition,
- cognitive maps and path finding,
- simulations of human imagery, and
- visualization for qualitative physical reasoning.

It seems a good guess that all of these problem areas share representations and algorithms. However, to date, most of them have evolved in different directions. Not all of these areas are covered here. In particular, visual object recognition is omitted entirely (see Vision).

A key issue in spatial reasoning is *qualitative shape representation*. Most humans have little trouble visualizing objects

and reasoning about them without precise knowledge about their dimensions. For instance, suppose a balloon landed on a pincushion. What might happen? Although the pins might puncture the balloon, one quickly realizes that they are unlikely to in this case because they are head-side-up. When most people solve a problem like this, they are obviously unaware of the exact shape of the pincushion or the detailed distribution of the pins. Yet they imagine a "picture" of the situation. Controversy has raged about what is really going on in the mind when this picturelike entity is experienced (see Imagery, below.) Fortunately for AI, the computational question can be asked how the knowledge about the shapes of objects like pins and pincushions is represented and used without an a priori commitment to any answer to questions about human visual imagery.

Various proposals have been made about representation of spatial information. Many of them limit themselves to two dimensions instead of three, either as a research tactic or because of a belief that it is desirable for efficiency to reduce three-dimensional problems to two dimensions when possible. Shape representations tend to fall into various categories: part-whole, volumetric, and surface descriptions (for an overview, see Ref. 1.)

**Part-Whole Descriptions.** Objects are described in terms of the parts that make them up. Typically these descriptions employ some kind of associative network. There is nothing special about this use of associative networks; the resulting descriptions would be similar to those in nonspatial domains, such as descriptions of corporate organizations. So the pincushion description might mention the presence of zero or more pins as parts.

The representation becomes more spatial when coordinates and other parameters are added to it. For instance, one might store with each pin its approximate length and position with respect to the pincushion. It is often useful to "invert" the resulting data structure. For instance, given a table of cities and their locations, it might be desirable to find a city near some location. Rather than search all the cities, one can use a discrimination tree (2), in which objects are sorted by discriminating on their X and Y coordinates (see Fig. 1). Other quantitative and symbolic discriminators can be introduced, such as the population or shape of each object. (The term "*k-d tree*" has been used in Ref. 3 for a tree discriminated on several numerical coordinates.) To find an object in such a tree, given as a key an X interval and a Y interval, the computer can start at the top and follow only branches compatible with the key, i.e., corresponding to subranges that overlap the key. Every leaf not reached contains cities located outside the key intervals, so only a subset of the cities are ultimately compared with the key interval.

**Volumetric Descriptions.** Objects are described as combinations of volumes. The volumes are often overlapping and often do not necessarily designate distinct parts of the overall shape. For example, a milk bottle might be described as a cylinder topped by a truncated cone. The dimensions and relative locations and orientations of the volumes must be specified (4,5). The component volumes may be drawn from a vocabulary of primitives or constructed by sweeping surfaces along axes [so-called generalized cylinders (qv)] (6). In some systems volumes may be subtracted as well as added. For example, a spool might be described as a solid cylinder with a small parallel

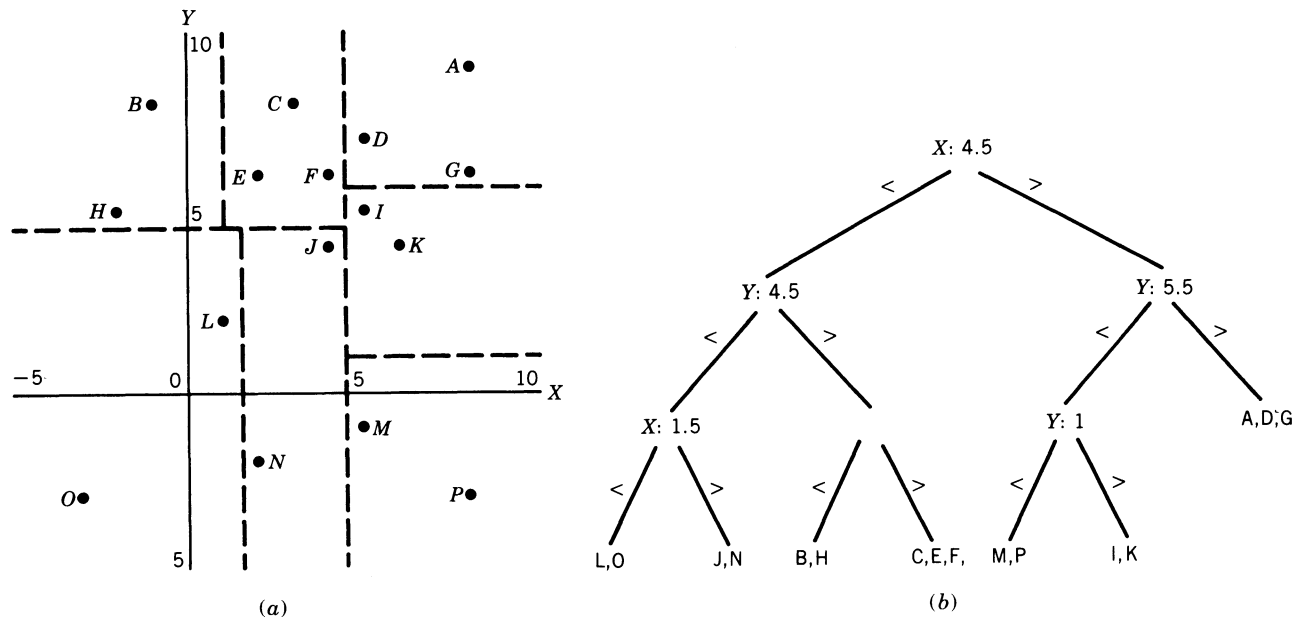


Figure 1. An X-Y discrimination tree: (a) cities; (b) discrimination tree.

cylinder subtracted from its axis. This approach is often called constructive solid geometry (7).

One of the simplest ways of describing volumes (or, in two dimensions, areas) is with arrays representing space (up to some grain), whose cells are labeled with the object filling that part of space. These are often called *occupancy arrays*, and their cells are called *pixels* in two dimensions (a term borrowed from computer graphics), or *voxels* in three dimensions. In Figure 2a the shape of a house is represented by putting an H in every cell the house fills. This is a two-dimensional projection; a three-dimensional array could be used if required. However, even the two-dimensional version is costly. The *quadtree* device allows the information to be compressed in an elegant way (8,9). The grid is represented as a tree whose nodes are square areas of the picture. The top node represents the entire picture. Each node has zero or four children. If the square region corresponding to a node lies entirely inside or outside the house, it is a leaf of the tree and is labeled with an H or E (for empty). Otherwise, it has four children representing the four quadrants of its square. The division stops at some convenient grain. See Figure 2b. The same idea applied to three dimensions yields the *octree*.

**Boundary Descriptions.** Objects are described by describing their bounding surfaces. The bounding surfaces are often planar, which limits the description to polyhedral objects or to polyhedral approximations of curved objects (10). In two dimensions the bounds are curves, often approximated as polygons or "polylines" (chains of lines, not necessarily closed curves) (11).

Many of these shape-representation ideas are borrowed from machine vision and graphics. If the goal is to draw a picture of an object, the application of these representations is well understood. Unfortunately, for the kind of spatial reasoning discussed above, much of the information provided in these formats is useless, and ideas have been lacking on what is needed instead. For instance, devices such as cubic splines are

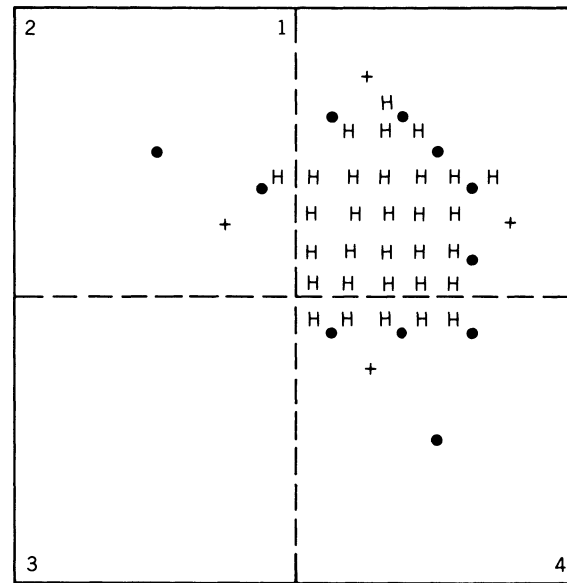
excellent for approximating a curve or surface to be drawn with just a few numbers. However, the resulting data compression is of little use in spatial reasoning because the numbers say little about the way the surface behaves. In spatial reasoning one is more likely to want information like *can be used as a conduit* or *is almost horizontal*. It is often necessary or desirable to approximate spatial knowledge. If a wall is almost flat, for many purposes there is no need to keep its slight deviations from flatness in mind, even if they are known. Hence the model used by the system for reasoning will usually be a simplified version of the truth. There may be multiple models for different purposes.

The most common approach to this problem is to use a simple symbolic vocabulary. If an object is known to be a cylinder, and nothing else is known about it, then it is described by the symbol "cylinder." Usually the machine knows something about the dimensions of the object, so it might be represented as

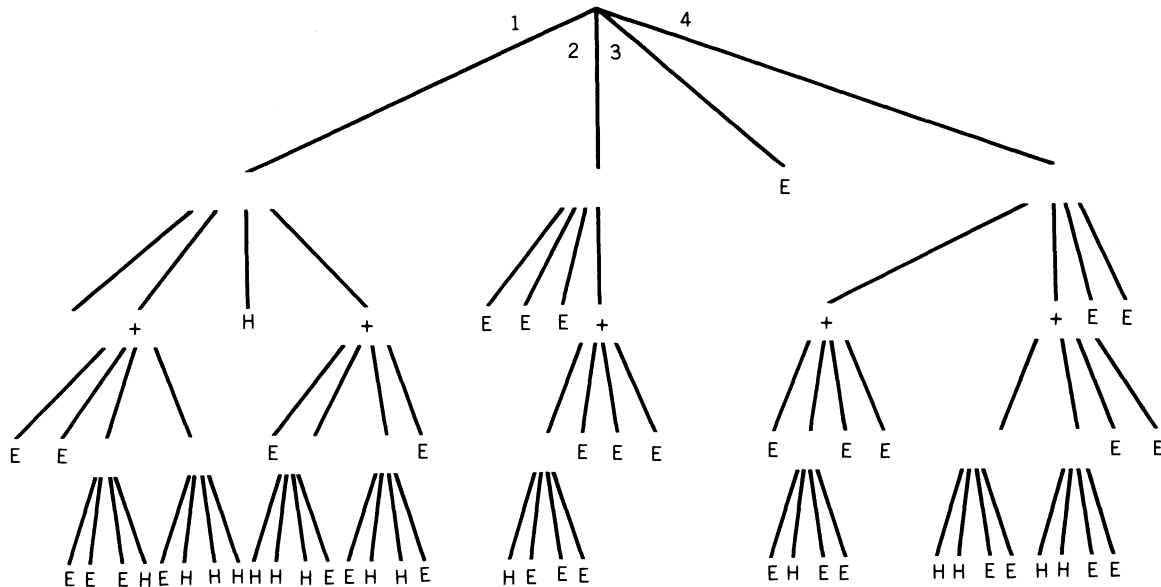
```

cylinder
length > width
axis-curvature = 0
  
```

(see Ref. 5 and above). You can think of this idea as sorting objects into qualitative "bins" with labels like cylinder within each bin objects are distinguished by different values of parameters like length and axis-curvature. This approach is by far the most common; for specialized applications, such "bins" are usually easy to find. For instance, in a route-finding application objects might be classified as streets, buildings, regions, rivers, etc., each with its own set of parameters. The boundaries between classes would seldom be crossed. The classes are useful because there is a large set of inferences that pertain to just the objects belonging to a given class. For instance, an object classified as a street can be used to get somewhere; an object classified as a river requires finding a bridge; etc.



(a)



(b)

Figure 2. Grid representation and quadtree: (a) occupancy array; (b) corresponding quadtree.

The difficulty with this scheme is generalizing it to handle more than one application area. In the general case, the following problems are encountered.

1. The qualitative-bin notation has trouble with detailed descriptions of objects. Once an object has been classified as a cylinder, it may be necessary to describe it further as *slightly flattened on one side* or *peppered with thousands of holes* (e.g., constructed of screening material). Little is known about how to turn such natural-language descriptions into something more formal.
2. An object may fall into more than one bin. The big advantage of the qualitative-bin idea is that similar objects have similar representations. The two objects in Figure 3a,b are obviously similar because they are both classed as cylinders

with different axis curvatures. But if the sequence of similar objects is continued, we ultimately arrive at Figure 3d, which would have been classed as a torus (with a gap, represented somehow). Such *qualitative discontinuities* may make it necessary to maintain multiple descriptions of objects.

3. An object may fall into no bins. For any given application it is usually easy to find qualitative classes that include every object of interest. It is much harder to find a set of classes that works just as well for every application. If you simply take all the classes that have ever been proposed, many objects will fall into more than one or none at all.

All these problems may be evidence that there is no solution to the "general spatial-reasoning problem." As elsewhere

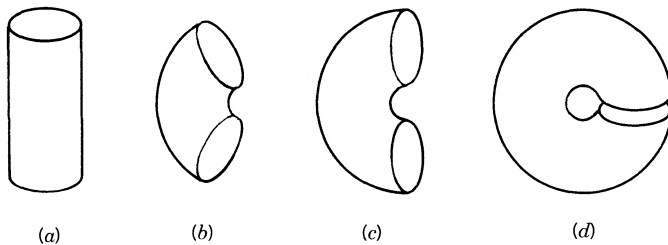


Figure 3. Object with multiple descriptions.

in AI, the mere ability to discern a problem may not mean it actually has a solution. Still, it is hard on introspective grounds to believe that humans have a collection of independent task-oriented spatial representations in their heads, and this is reason to keep looking for a solution to the general case.

In what follows, various specialized problem areas are examined, keeping some of these problems in mind.

### Route Finding and Exploration

The route-finding problem is planning a route from one place to another and then following it. It is assumed that the planner (henceforth referred to as the "robot") has a *cognitive map* of its surroundings that it can consult for this purpose. This map is incomplete, so that the planner may need to ask directions or explore as it goes. So far robots have not evolved to the point where they could actually carry out such plans, so simulated robots have been used instead. In this section it is assumed that the robot is small compared to the space through which it is navigating. A somewhat different line of research assumes that the robot is large but that the shapes and positions of all obstacles in the space are known so that intricate reasoning is required to squeeze it through. This is called the robot motion-planning problem (see Motion analysis).

If it is assumed that the robot is navigating through a path network, such as the streets of a city, algorithms developed by Kuipers (12) and Elliot and Lesk (13) are appropriate. Elliot and Lesk's algorithm is designed to provide automatic directions to places in a city. The algorithm is given a complete street map of the city, including the map coordinates of every intersection. Given a starting intersection and a desired destination, it finds a near-optimal route between them. It operates by doing a heuristic search from the starting intersection using the global coordinates to keep it on track. The heuristic

evaluation function favors large streets and penalizes routes containing many left turns so that the directions it finds tend to be the sort humans like to follow. The algorithm works rather well.

Kuipers' program is intended to be closer to a model of how people follow directions and learn from following them. It stores a path network, but it is incomplete, and there are no global coordinates stored for anything. It also maintains a hierarchical structure of regions. Paths are located inside regions and can run between regions. This structure makes it natural to find hierarchical plans for getting from one place to another (a *hierarchical plan* for a task consists of a short sequence of large steps, each of which is broken down into smaller plans if necessary.) Here is a sketch of an algorithm for finding such plans (see Figure 4):

To find a path between two points  $P_1$  and  $P_2$

Find the smallest region  $R$  including both points

Find the regions  $R_1$  and  $R_2$  just below  $R$  that contain  $P_1$  and  $P_2$  respectively

Find a route  $Q_{12}$  from  $R_1$  to  $R_2$  (by some search process)

Recursively find a route  $Q_1$  from  $P_1$  to  $Q_{12}$  in  $R_1$  and a route  $Q_2$  from  $P_2$  to  $Q_{12}$  in  $R_2$

Return result  $Q_1-Q_{12}-Q_2$

This algorithm is not guaranteed to return optimal routes, but it works without requiring any sort of global coordinates. All it requires is that the area actually be organizable into the appropriate hierarchical structure of major routes between fairly well-defined regions.

McDermott and Davis have looked at the route-finding problem from a somewhat more general perspective in which objects other than paths are represented (2,11). This perspective requires them to use a more general representation for the shapes and locations of objects. In one representation objects are represented as rectangles or circles (2); in another they are represented as arbitrary polygons (11). In each case an important component of the representation is numerical parameters (point coordinates or lengths of lines). To capture ignorance, these numbers are stored as intervals (assimilating new information into the database can cause these intervals to become narrower; for instance, one might discover that an object whose length had been known to lie in the interval 10–50 m was actually about 25 m long, so that its length lies in the interval 23–26 m). In addition, objects can be stored in a discrimination tree (see above).

The route-finding problem reduces to several special cases in this representation. Using a hierarchical region representation like Kuipers's, one can determine whether the problem requires traversing mainly an open area with obstacles (like a field with boulders) or a congested area with conduits (like a city). In either case the algorithm uses the discrimination tree to find the largest obstacles or conduits, and it outputs a plan at a high level of generality first. Hence, in Figure 5 the highest level of the plan deals with getting across the river; steps at that level get expanded into plans for getting to the bridge and then to the destination from the far side.

Where do cognitive maps come from? One source is from verbal material, such as descriptions of routes (12,14). Another source is from the map-owner's eyes as he or she wanders

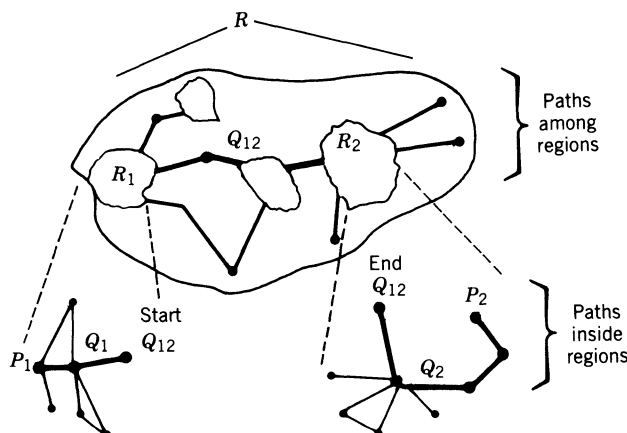


Figure 4. Finding routes between points in different regions.



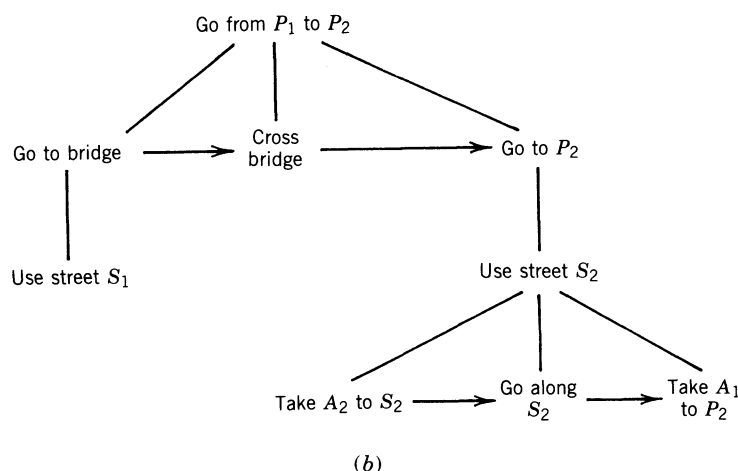
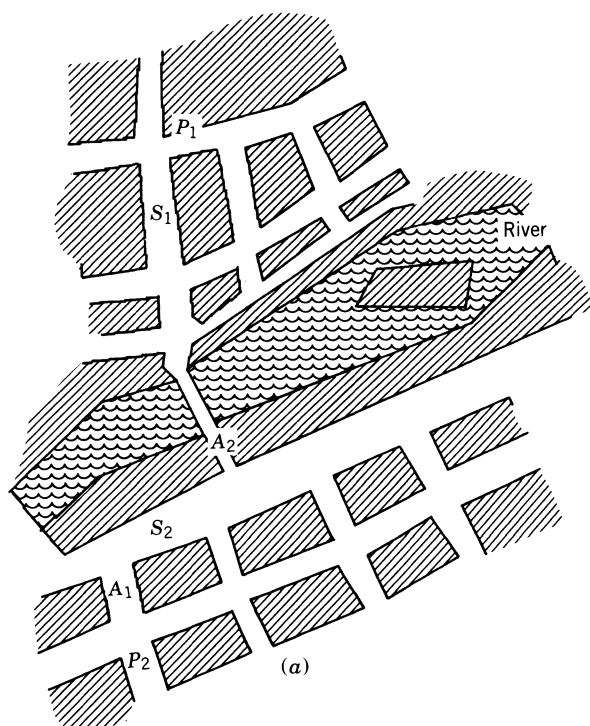


Figure 5. Map and resulting navigation plan.

about. Davis (11) proposed one model for how visual assimilation happens. He assumed that low-level vision problems were solved, that the visual system could deliver to the spatial reasoner a map of its immediate surroundings. The problem then is to integrate this map into the existing one, recognizing old objects (and extending and sharpening one's knowledge of them) and making new entries for unfamiliar objects. Since objects are stored as approximations, there must be a matching algorithm that can decide when two descriptions could be of (possibly overlapping parts of) the same object. Davis proposed one such algorithm for polygonal approximations; Lavin (15) proposed one for a more specialized representation of Gaussian hills.

### Imagery

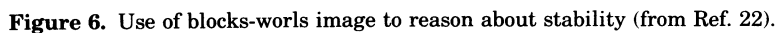
When people solve spatial-reasoning problems in their heads, they often have a subjective feeling of seeing a picture. For example, if asked to name all 50 states, almost everyone reports visualizing a map through which the attention wanders. There is a controversy about what these reports are reports of. On one side are researchers like Shepard (16) and Kosslyn (17,18), who believe that there is an actual picturelike entity in the brain performing useful computations. On the other are critics like Pylyshyn (19,20), who believe that pictures are poor computing devices and that subjective impressions are misleading.

The first issue to settle in building a computational model of imagery is what the underlying picture medium is. You might begin by assuming that a map of the United States was stored as a hierarchical pointer structure, with nodes representing large areas like New England pointing to smaller component areas like states, and other, less familiar areas pointing to a sparser set of subareas. Such a model might explain many facts [e.g., certain distortions in subjects' memories of maps (21)] but would not by itself explain why the data struc-

ture is experienced as a picture when it is traversed. Those who feel that this is one of the prime facts to be explained make their models pictorial from the start.

A dangerous pitfall here is to assume that a mental image is nothing but a picture, so that some homunculus must "look" at it. To avoid the need for the homunculus, imagists usually assume that the pictorial medium is active, capable of computation on its own. They assume that the picture consists of arrays of pixels as in occupancy arrays, except that each is capable not just of holding a small patch of the picture but also of performing simple computations, including communication with its neighbors. Interesting inferences then occur by the combined action of all the pixels. In the brain such pixels are thought to correspond to groups of neurons, probably laid out in an actual two-dimensional field. In a digital computer each pixel is a data structure arranged in an array, and a sequential algorithm simulates the parallel communication. There are two attractive qualities to such a model. One is the parallelism, which, as usual, promises fast computation. The other is the potential for using images to discover "new" relationships among entities. The machine can make up a picture of some entities using one set of relationships and then "examine" the picture to discover some new relationships. The hard part is to find applications where these qualities are actually realized.

One example is Funt's program WHISPER (22), which solves problems in the "blocks world" by using an image of this kind, as exemplified by Figure 6. Block A is indicated by labeling the pixels it occupies with an A for pixels in A's interior, or a 1 for pixels on A's boundary. In that figure the program decides that block B is unstable and will cause D to fall when it does. The image machine helps in several ways. First, the center of gravity of B must be found, by having each pixel labeled B or 2 report its location to a central processor, which adds the coordinates and averages. The processor notes that the center of gravity of B is not above A, so B will fall. Then it



simulates the fall by instructing each pixel of B to rotate around the upper right corner of A. This rotation occurs step by step (actually, two arrays of pixels are used; consult Ref. 22 for details). The collision between B and D is noticed when a 2-labeled pixel attempts to transfer its contents into one already labeled with a 4. The advantage of this approach to thinking about colliding blocks is that no intricate calculation about intersecting lines is required to detect a collision.

Kosslyn and Schwartz's program (23) is intended more to be faithful to psychological data than to perform well on a task. It mimics the way humans report constructing images of objects such as automobiles at different scales and different levels of detail.

### Physical Reasoning

An active area of AI research is the study of reasoning about the structure and function of physical systems. Here is where one would expect to find the consumers of the output of spatial-reasoning research. For instance, Figure 7 shows a problem solved by de Kleer's program NEWTON (24,25). The program reasons about the qualitative shape of the roller coaster in order to realize that the object cannot get to point X. Other physics-problem solvers reason about situations of similar complexity (26,27).

Unfortunately, none of these programs rely on a general-purpose shape representation. All of them use some version of the qualitative-bin representation. For example, Novak's program accepts physics-problem statements in natural language. It turns a sentence like "a man stands on a ladder" into an internal representation in which a man modeled as a point mass is located on a ladder represented as a line. Each such internal entity has various parameters associated with it, such as the man's mass. In different problem statements the man would get modeled as some other kind of entity, with different parameters. Equations involving these parameters are set up and solved to produce the solution to the overall problem.

More recently there has been a lot of work on qualitative analysis of physical systems (28,29), a typical such system being a kettle of water on a stove. The result of the analysis is called an *envisionment* and is an account of the ways in which the system may behave given its initial state. In spite of the use of this term, the spatial representations in such programs are no more complex than their predecessors, typically consisting of associative networks describing the components of the

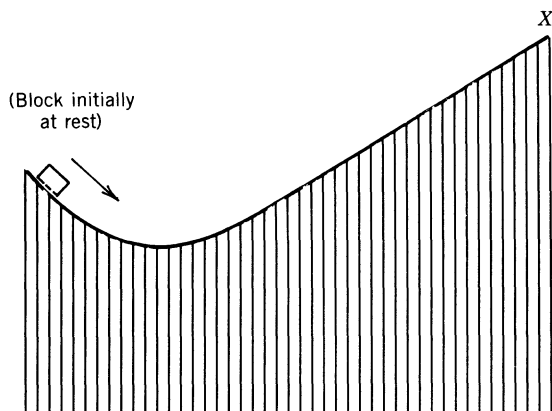


Figure 7. Problem solved by NEWTON, "Will the block reach point X?" (from Ref. 24).

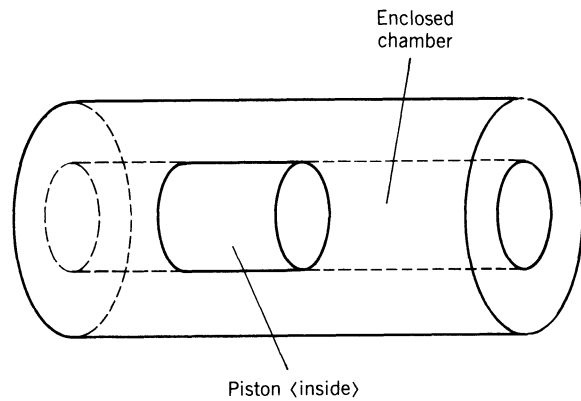


Figure 8. Simple pneumatic machine (from Ref. 30).

system and their interactions. An exception is the work of Stanfill (30). His program takes a description of the parts of a simple pneumatic machine and their layout and figures out what the machine does. The description uses a general vocabulary of volume combinations (see above). The system infers the relative degrees of freedom of the parts and notes connected volumes containing gas. It then combines these descriptions in order to infer a model of the qualitative behavior of the machine. For instance, for the piston-cylinder machine of Figure 8, it would decide that the acceleration of the piston is entirely determined by the pressure difference between the enclosed chamber and the earth's atmosphere. This conclusion could then be used as a starting point for qualitative envisionment.

Another exception is in Forbus (31), where the FROB program for reasoning about the behavior of a bouncing ball is described. Given a map of a situation, through which one or more balls were launched, it can answer questions like: where will the ball wind up? The map is given as a two-dimensional line drawing with the coordinates of all points known. However, the program redescribes it as a graph of regions with different properties so that it can reason qualitatively about the trajectories of balls flying and bouncing through them.

Finally, there is the work of Hayes (32,33) on "naive physics" (qv), the effort to formalize what "everyone knows" about the way objects in the world behave and interact. He presents axioms about the behavior of liquids in containers, where the containers are described in terms of their bounding surfaces. The notation omits most of the details of where these surfaces are located or how they are shaped and instead focuses on qualitative description of them as separating volumes into different functional parts. An open container, for instance, is described as a volume with just one free face, the top. The brim of the container is the "face of its top face," that is, the edge bounding its top face. Hayes argues that the way to bring time and change into this formalism is to analyze activity as four-dimensional *histories* of objects. For example, if an open container full of liquid is tilted, there will be a "leaving" history at the brim of the container that interfaces to a "falling" history in the free space nearby.

### BIBLIOGRAPHY

1. D. Ballard and C. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
2. D. V. McDermott and E. Davis, "Planning routes through uncertain territory," *Artif. Intell.* **22**, 107-156 (1984).

3. J. Bentley and J. H. Friedman, "Data structures for range searching," *Comput. Surv.* **11**, 397–409 (1979).
4. D. Marr and H. K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," *Proc. Roy. Soc. B* **200**, 269–294 (1978).
5. R. A. Brooks, "Symbolic reasoning among 3-D models and 2-D images," *Artif. Intell.* **17**, 385–348 (1981).
6. G. J. Agin and T. O. Binford, "Computer descriptions of curved objects," *IEEE Trans. Comput.* **C-25**, 439–449 (1976).
7. A. A. G. Requicha, "Representations of rigid solid objects," *Comput. Surv.* **12**, 437–464 (1980).
8. A. Klinger and M. L. Rhodes, "Organization and access of image data by areas," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-1**, 50–60 (1979).
9. H. Samet, "Region representation: Quadrees from boundary codes," *CACM* **23**, 163–170 (1980).
10. B. G. Baumgart, Geometric Modeling for Computer Vision, Stanford Artificial Intelligence Lab Memo STAN-CS-74-463, 1975.
11. E. Davis, Representing and Acquiring Geographic Knowledge, Yale University Computer Science Department TR 292, 1984.
12. B. Kuipers, "Modeling spatial knowledge," *Cog. Sci.* **2**, 129–154 (1978).
13. R. J. Elliot and M. E. Lesk, Route Finding in Street Maps by Computers and People, *Proceedings of the Second AAAI Symposium*, Pittsburgh, PA, pp. 258–261, 1982.
14. C. K. Riesbeck, "You can't miss it: Judging the clarity of directions," *Cog. Sci.* **4**, 285–303 (1980).
15. M. A. Lavin, Analysis of Scenes from a Moving Viewpoint, in P. H. Winston and R. H. Brown (eds.), *Artificial Intelligence: an MIT Perspective*, Vol. 2, pp. 185–208, 1979.
16. R. N. Shepard and J. Metzler, "Mental rotation of three-dimensional objects," *Science* **171**, 701–703 (1971).
17. S. Kosslyn, "Information representation in visual images," *Cog. Psychol.* **7**, 341–370 (1975).
18. S. Kosslyn, *Image and Mind*, Harvard University Press, Cambridge, MA, 1980.
19. Z. W. Pylyshyn, "What the mind's eye tells the mind's brain," *Psychol. Bull.* **80**, 1–24 (1973).
20. Z. W. Pylyshyn, *Computation and Cognition: Toward a Foundation for Cognitive Science*, MIT Press, Cambridge, MA, 1984.
21. A. Stevens and P. Coupe, "Distortions in judged spatial relations," *Cog. Psychol.* **10**, 422–437 (1978).
22. B. V. Funt, WHISPER: A Computer Implementation Using Analogues in Reasoning, Report 76-09, University of British Columbia, Department of Computer Science, 1976.
23. S. Kosslyn and S. Schwartz, "A simulation of visual imagery," *Cog. Sci.* **3**, 265–295 (1977).
24. J. de Kleer, Qualitative and Quantitative Reasoning in Classical Mechanics, Report 352, MIT AI Lab, Cambridge, MA, 1975.
25. J. de Kleer, Qualitative and Quantitative Reasoning in Classical Mechanics, in P. H. Winston and R. H. Brown (eds.), *Artificial Intelligence: an MIT Perspective*, Vol. 1, MIT Press, Cambridge, MA, pp. 11–30, 1979.
26. G. S. Novak, Representations of Knowledge in a Program for Solving Physics Problems, *Proceedings of the Fifth IJCAI*, Cambridge, MA, pp. 286–291, 1977.
27. A. Bundy, "Will it reach the top? Prediction in the mechanics world," *Artif. Intell.* **10**, 129–146 (1978).
28. J. de Kleer and J. S. Brown, "A qualitative physics based on confluences," in J. Hobbs and R. C. Moore (eds.), *Formal Theories of Commonsense World*, Ablex, Norwood, NJ, 109–183, 1985.
29. K. D. Forbus, "Qualitative process theory," *Artif. Intell.* **24**, 85–168 (1984).
30. C. Stanfill, The Decomposition of a Large Domain: Reasoning about Machines, *Proceedings of the Third AAAI Symposium*, Washington, DC, pp. 387–390, 1983.
31. K. D. Forbus, A Study of Qualitative and Geometric Knowledge in Reasoning about Motion, TR 615, MIT AI Lab, Cambridge, MA, 1981.
32. P. J. Hayes, The Second Naive Physics Manifesto, in J. Hobbs and R. C. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, New Jersey, pp. 1–36, 1985.
33. P. J. Hayes, Naive Physics 1: Ontology for Liquids, in J. Hobbs and R. C. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, New Jersey, pp. 71–107, 1985.

D. V. McDERMOTT  
Yale University

## REASONING, TEMPORAL

It is hard to think of a research area within AI that does not involve reasoning about time in one way or another: Medical-diagnosis systems try to determine the time at which the virus infected the blood system; circuit-debugging programs must reason about the period over which the charge in the capacitor increased; automatic programmers and program synthesizers must deduce that after procedure *P* is executed the value of variable *X* is zero; and robot programmers must make sure that the robot meets various deadlines when carrying out a set of tasks. One particular subfield of AI, which has become known as the area of temporal reasoning (TR), acknowledges this central role. Although most subfields merely employ temporal terminology, the very goal of TR is a general theory of time.

Of course, the passage of time is important only because changes are possible. In a world where no changes were possible—no viruses infecting blood systems, no electrical charges changing, no changes in program counters, not even changes in the position of the sun in the sky or the position of the hands on wristwatches—not only would there be no computational justification for keeping track of time but the very concept of time would become meaningless. Therefore, the ideal theory of time must meet two requirements. The first is that it provide a language for describing what is true and what is false over time. The second is that it provide a criterion of "lawful change."

The work that is described here is mostly in the form of various "formalisms," which usually means a logic with more or less well worked-out syntax and semantics. The first section reviews the work done within AI, describing a somewhat idealized progression of such formalisms. The last section describes related work from philosophy and theoretical computer science.

How does one represent temporal information? Suppose one wants to represent in logic the fact that the color of a particular house is red at time *t*. One has several options:

Time can simply be included as an argument to the predicate: COLOR(HOUSE,RED,*T*), where *T* is a time argument (a point, an interval, or otherwise; see later discussion).

The proposition can be "reified" (or, as John McCarthy once proposed, "thingified"): HOLD(*T*,COLOR(HOUSE,RED)).

Here COLOR serves as a function symbol rather than a relation symbol.

Time need not be mentioned at all! Instead, one complicates the interpretation of formulas. If in classical logic a formula  $\varphi$  is either true (written  $\models \varphi$ ) or false (and in this entry the discussion is confined to the propositional case), now a formula is either true at a given time  $t$  (written  $t \models \varphi$ ) or false at that time. (Here again one has of the choice of choosing  $t$  to be a point, an interval, or some other temporal entity.)

The first option is not acceptable from the standpoint of TR. If time is represented as an argument (or several arguments) to predicates, there is nothing general you can say about it. For example, you cannot say that "effects cannot precede their causes"; at most you can say that about specific causes and effects. Indeed, this first option accords no special status to time—neither conceptual nor notational—which goes against the grain of the TR spirit.

The second option fares better in this respect, and in one guise or another has been widely accepted in TR. Taking this approach seriously requires paying special attention to the meaning of the terms in the language, which now also serve the function of what otherwise would be relation symbols.

The third option, favored in modern philosophy and theoretical computer science, has for the most part been ignored in AI until very recently.

### Temporal Reasoning in AI

**Situation Calculus, STRIPS, Histories, Intervals, and Chronicles.** McCarthy and Hayes introduced the situation calculus (SC) (1), a temporal formalism that to this day is the basis for many temporal representations. A situation is a snapshot of the universe at given moment. Actions are the means of transforming one situation into another. For example, by performing the action PICKUP(A) in the situation where ON(A, B) and ISCLEAR(A) are true, one arrives at a new situation where ISCLEAR(B) is true. The actual formal construction uses the function RESULT that accepts a situation and an action as arguments and returns a new situation. If in the above example the first situation is S1 and the second S2, then  $S2 = \text{RESULT}(S1, \text{PICKUP}(A))$ . One can of course construct longer chains of action, as in  $S3 = \text{RESULT}(\text{RESULT}(S1, \text{PICKUP}(B)), \text{PUTDOWN}(B))$ .

SC makes several strong commitments. The first is about discreteness of time, which precludes discussion of continuous processes such as water flowing into a container and gradually filling it. The second is about contiguity of cause and effect, so to speak; the effects of an action are manifested at the very next situation. A further limitation of SC was that it did not allow concurrent actions, even in the framework of discrete time. The best known problem introduced by SC is the frame problem. Consider the same situations S1 and S2 as described above with the additional information that in S1 COLOR(A, GREEN) is true. Is COLOR(A, GREEN) still true in S2? One would like to answer affirmatively, but in fact this conclusion is not warranted by the theory. The RESULT function specifies only what changes as a consequence of taking an action, but not what is true by virtue of having not changed. In order to be able to make those inferences, one can add numerous frame axioms, specifying for each action what it does not

affect. The first problem with this solution is that such axioms are numerous: Picking up a block does not change its color, does not affect any other block, does not change the president of the United States, etc. It is obviously impossible to explicitly list what is unaffected by an action. A further complication arises if one introduces concurrent actions. In this case frame axioms are simply wrong: Someone might paint the block as it is being PICKUPed.

As was mentioned before, despite these limitations SC has proved very influential. For example, it has been the basis for several planning (qv) systems. One of the first of such systems was STRIPS (2), which embodied a natural solution to the frame problem. STRIPS is a name for both a formalism and a planner based on that formalism. This entry is concerned primarily with the former. The STRIPS framework adopted the same view of time as SC but made the following addition. With each action STRIPS associated two lists. The addlist specified what becomes true as a result of the action, and the deletelist specified what ceases to be true after the action. The STRIPS assumption was that if action A transformed state S1 into state S2, then a proposition  $P$  was true in state S2 if and only if either  $P$  was in the addlist of A or  $P$  was true in S1 and was not in the deletelist of A. This assumption was the basis for the regression operator in the STRIPS planner (see Planning).

One of the strong advocates of formal reasoning about the commonsense world has been Hayes. In "The Naive Physics Manifesto" (3) (see Physics, naive) he offers general justification of his approach, and in Ref. 4 he offers an actual formalism to describe the behavior of liquids. (Although these are recent publications of slightly revised versions, the original papers were written in the seventies.) In the latter paper Hayes introduced the notion of histories, which has had a strong influence on TR. A history is a connected piece of four-dimensional space-time. For example, a "falling" history is the space occupied by a liquid for the duration of its free fall. This view of the world is a radical departure from the SC paradigm. It acknowledges the continuous nature of time (and space, although that is not directly relevant to this discussion) and allows the representation of gradual change. One is not restricted to snapshots of the universe and a method for stringing them together. Instead one describes an entire interval of time. Of course, one can describe a snapshot of the universe by taking a slice through a history (which is the projection of the history onto the three-dimensional space at a given point in time). The theory of histories was partially applied by Forbus to reasoning about qualitative physics (5).

The spatial nature of Hayes's histories has drawn some criticism. For one thing, some occurrences do not have a well-defined spatial extent: What are the spatial boundaries of a conversation? Of an election? Of a confusion? Furthermore, histories are extensional in that the space-time of the history completely determines the history. But this cannot possibly be right, due to the fusion problem: More than one "history" can take place at the same space-time, e.g., a concert history and a stale-air history. Nevertheless, Hayes's bold transition from states to interval inspired much of the later work, including Allen's interval calculus and McDermott's temporal logic.

Retaining the interval-based view of time, Allen proposed a theory of action and time. In Ref. 6 he identifies the 13 possible relations between two intervals (identity, the one totally preceding the other, overlapping, etc.).

The ontology proposed by Allen associates with an interval

one of three objects: a property, an event, or a process. A property is something that is "statically" true or false; e.g., "the pen is red." Allen represents this assertion by the formula  $\text{HOLD}(I, \text{COLOR}(\text{PEN}, \text{RED}))$ . Properties hold uniformly throughout an interval; a proposition holds for an interval exactly when it holds for all subintervals:  $\forall I, P \text{ HOLD}(I, P) \equiv \forall I' \in I \text{ HOLD}(I', P)$  (and in fact Allen requires a slightly stronger axiom). Since properties are "reified" propositions, in order to retain their intuitive meaning, Allen simulates the logical connectives. For example,  $\text{HOLDS}(I, \text{AND}(P, Q)) \equiv \text{HOLDS}(I, P) \wedge \text{HOLDS}(I, Q)$  is an axiom in Allen's system.

Contrary to properties, events are wholistic entities. If an event occurred over an interval, it did not occur over any subinterval. An example of an event is "I went to the shop"; such an event is repeatable but not divisible. The predicate denoting occurrence of events is  $\text{OCCUR}(I, \text{EVENT})$ . Processes are a hybrid case. An example of a statement describing a process is "I am walking"; if it is true for an interval, it must be true for some subinterval but need not be true for all of subintervals (I may rest during my hour-long walk).

Allen's proposal also include an account of causation. "Event causation" describes a relation between events (and associated intervals). The properties of this relation are exactly the following: The occurrence of causes entails the occurrence of effects, effects may not precede their causes, and the relation is transitive and antisymmetric. Other parts of Allen's proposal include the notions of agents, actions, level-generation (7), intentions, plans, commitment, knowledge, and belief.

McDermott (8) began exploring the connection between problem-solving (qv) and theories of time and action. He proposed a temporal logic to be used in the process of planning (9), in which he introduced the notion of chronicles. McDermott's construction takes as primitive the notion of a state, which is a point in time in some set of possible worlds. A fact-type is a reified proposition that may be instantiated as a fact-token. For example, "I am walking" is a fact-type, and "I am walking on 1.1.2000 from 1:00 to 1:45" is a fact-token. The construction is set-theoretic: the assertion  $T(S, P)$ , denoting the existence of a particular fact-token, is shorthand for  $S \in P$ . Similarly for an event-type  $E$ , the assertion  $\text{OCC}(S1, S2, E)$ , denoting the existence of an event-token, is merely shorthand for  $\langle S1, S2 \rangle \in E$ . The assertion  $\text{TT}(S1, S2, P)$  (" $p$  is true throughout the interval  $[s1, s2]$ ") is shorthand for  $[S1, S2] \subset P$ .

McDermott assumes that time, as well as being a partial order, is dense. He furthermore arranges time in chronicles, which are linear time-lines that form a treelike structure: Chronicles may branch into the future, and once they do, the different branches do not meet again. When one contemplates taking an action, one needs to compare the world (or chronicle) in which the action is taken to the world in which it is not. McDermott's formulation was the first one to provide a mechanism for such a comparison within the logic; different worlds are simply separate paths in the chronicle tree. In particular, he managed to give meaning to the notion of preventing.

McDermott also devotes a large part of the discussion to causation as a representation of rules governing change. The notion he considers are ECAUSE, for one event causing another, and PCAUSE, for an event causing a fact. The former allow for a delay between the cause and the effect. The latter are particularly interesting, since they introduce the notion of persistences. The idea is that the effect of a PCAUSE is a defeasable prediction. For example, the predicted effect of a

boulder rolling to a particular location is that the boulder will be there for the next 50 years. This prediction will be violated if a construction company decides to erect a building on that particular site at an earlier time. Persistences, whose semantics are loosely based on nonmonotonic logic (see Reasoning, nonmonotonic) (10), are McDermott's solution to the frame problem. Other topics covered by McDermott are continuous change, qualitative physics and potrans (potential transfer), planning, and the notion of a subtask. Further application of temporal reasoning to planning can be found in Ref. 11.

**Applications to Qualitative Physics.** Much of physical reasoning involves time for obvious reasons. One can reason about what a physical system will do starting from initial conditions, or contrariwise one can reason backward from final conditions to an explanation in terms of previous behavior. Most of the work has focused on the former process, which is called envisioning (12) or projection (13) (but see Ref. 14).

Envisioning is not the same as simulation. One wants the reasoner to see interesting patterns in the behavior, to recognize an oscillator, not just become one. Furthermore, one wants the reasoning to be able to proceed in the absence of the detailed quantitative information that a simulator would require. The resulting paradigm is called qualitative reasoning about physical systems. It has been pursued by de Kleer and Brown (15), Forbus (5), and Kuipers (16) (see Physics, naive; Reasoning, causal; Reasoning, common sense).

There are two phases to the reasoning in this paradigm. First the program must produce an abstract version of the structure of the system; then it must use this structure to reason. In almost every case, the abstract structure is analyzed as a set of devices with connections to each other. "Causality flows" through the connections from one device to another. The state of a device is represented by the values of a set of quantities. A device can have different behaviors depending on what ranges various quantities lie in. For instance, a thermostat-controlled heating system will be characterized by quantities like room temperature and heat flow out of room. The furnace will be in state OPERATIVE or OFF depending on whether the room temperature is greater than the thermostat setting. If it is OPERATIVE and the tank is NONEMPTY, there will be a constant value of furnace heat flow into the room. And so forth.

Devices "connect quantities together"; they enforce certain relationships among them. Hence, from an initial set of quantity relationships, the temporal reasoner can deduce how the quantities will change. If the furnace is on, and it is not too cold outside, the room temperature will increase. Any such change, if allowed to continue, will eventually drive some quantity out of its current range, thus changing the behavior of some device. The reasoner must therefore reproject the behavior of the system under the altered circumstances. The new analysis will result in a different pattern of changes, leading to new behavior shifts, and so forth. The final analysis may be displayed as a graph, as in Figure 1. States of the system are shown as ovals. An arrow joins two states if the first may evolve into the second through quantity changes. A loop in the graph indicates that the system may oscillate between the two given states. If the analysis is correct and useful, one can expect to find a finite number of significantly different states and hence expect every system to enter one state and stay there or loop among some set of states. Note that the "furnace on" state has two potential successors: either the room be-



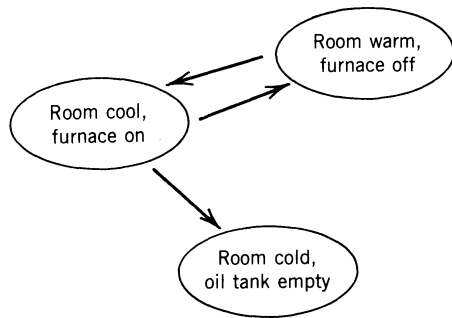


Figure 1. A simple state diagram.

comes warm or the oil runs out. Both possibilities appear because the qualitative analysis shows two quantities changing: the room temperature is increasing and the oil level is decreasing. Without further information about the magnitudes of the quantities, the reasoner cannot decide which will reach its threshold first and cause the system to change state.

The early work by Hendrix (17) anticipated much of current research in planning and qualitative physics. Hendrix's extension of the STRIPS formalism includes representation of continuous change (using real values for quantities) and concurrent actions. The central component of the system (which is a set of data structures and a skeleton of a simulation program rather than a logical formalism) is the process monitor. This module continuously attempts to identify "active processes" and compute their effects.

Rieger also addresses the issue of continuous change in his proposal of commonsense algorithms (18). Among the many notions he considers are continuous causality and gated causal rules. His system too is couched in terms of data structures and algorithms, and McDermott (9) pointed out some of the difficulties in assigning meaning to the symbols in Rieger's system.

Another system that is primarily a serious attempt at incorporating one of the more sophisticated logics into a computer program is Dean's Time Map Managing System (19). The system, which can be viewed as a temporal reason-maintenance system, was designed to be used as part of an automated planner. It is loosely constructed around McDermott's temporal logic. A earlier attempt along similar although more modest lines is Vilain's system (20), a time-maintenance system based on a formalism that is similar to Allen's.

It is worth pointing out that the more "applied" work on TR has not been based to any great extent on the representational research described earlier. One can view this as evidence that the foundations of the applied work are shaky, or that the logic-level research has not yet become sophisticated enough to be of real value, or both.

### Work Outside AI

Both AI and theoretical computer science owe an intellectual debt to philosophy, where time, action, and causation have been studied for many years. Although it is not possible here to properly cover the relevant philosophical literature, two good expositions of the work done on causation are Refs. 21 and 22. For recent investigations into the nature of actions see Refs. 7 and 23.

Work done in formal philosophy on the logic of time is par-

ticularly relevant. It was mentioned in the introduction that one possibility for representing temporal information is to have time implicit in the interpretation of the formulas. The way this option is exercised is through modal logic, to which a modern introduction is Ref. 24. The formulas of the logic are augmented by one or more modal operators; if  $P$  is a wff and  $\Box$  is a (unary) modal operator, then  $\Box P$  is a wff too. In the basic modal logic there is a single modal operator  $\Box$ , called "box" and pronounced "necessarily." Its dual operator  $\Diamond$ , called "diamond" and pronounced "possibly," is defined by  $\Diamond P \equiv \neg \Box \neg P$ . The widely accepted semantics for the resulting logic are possible world semantics introduced by Kripke (25). When applied to temporal logic, possible worlds are equated with time points, and the modal operators are usually some variant of the following:

- FP:  $p$  is true in some future time point;
- GP:  $p$  is true in all future time points;
- PP:  $p$  is true in some past time point; and
- HP:  $p$  is true in all past time points.

The outstanding feature of these systems is the indexicality of time: formulas are interpreted with respect to a time point (usually called now) and may contain reference to other time points through use of the modal operators. Thus, if you take  $p$  to mean "it is raining," the formula  $\neg p \supset G \neg p$  means "if it isn't raining now then it never will." From this logic one can derive a more traditional modal logic such as S4, for example by defining  $\Box p$  to be  $p \wedge Gp$  (The other well-known construction is define  $\Box p$  to be  $Hp \wedge p \wedge Gp$ .)

Prior is the philosopher widely credited for first applying the principles of modal logic to temporal logic (see, e.g., Ref. 26). He explores various possible properties of time—linearity, future branching, circularity, additivity, having a metric defined on it—and relates the resulting logics to known modal systems. Rescher and Urquhart (27) do the same, employing more conventional notation and using more recent results from logic. In particular, they discuss decision procedures based on the semantic tableau method. The most recent comprehensive study of temporal logics appears in a book by van Benthem (28). He conveniently divides the discussion into the structure of time on the one hand and the nature of temporal assertions on the other. In each part he considers two cases: basing the logic on points and basing the logic on intervals, the latter possibility having recently gained currency in philosophy. As was said earlier, the philosophical literature on time is very rich; these references are merely initial pointers to it.

In theoretical computer science there has been considerable interest in TR, although until recently there was no overlap between that work and TR in AI. Pnueli was the first in computer science to apply modal temporal logic to program verification (29,30). There are several variants of modal temporal logic, stemming from different models of time (discrete or continuous, linear or branching) and different choices of modal operators (cf. Ref. 31). Discreteness is usually assumed, and "time points" correspond to the instants when the program interpreter is about to execute the next command. The modal operator  $\Box$  is the "next-state" operator:  $\Box P$  is true iff  $P$  is true the next time the interpreter is about to execute a command. Using the logic, various properties can be expressed very concisely, such as termination, freedom from deadlock, "fair execution," and more (32). Temporal logic has been the frame-

work in which much research on concurrent computation was done (see Ref. 33).

Dynamic logic, first introduced by Pratt (34) in conjunction with Moore, is the other major area of TR in theoretical computer science, and it too is geared toward reasoning about computer programs and digital devices. Rather than the usual modal operator of temporal logic, dynamic logic associates modal operators with each program. If  $\alpha$  is a (nondeterministic) program, then  $\langle\alpha\rangle P$  means that  $P$  holds after some possible execution of  $\alpha$ . Similarly,  $[\alpha]P$  means that  $P$  holds after all possible executions of  $\alpha$ . For a systematic treatment of dynamic logic, from so-called simple dynamic logic to the full-fledged first-order one, see Ref. 35.

Both dynamic logic and temporal logic interpret statements over time points. Since in AI one often encounters statements that refer to time intervals rather than time points ("the robot performed the task," "I solved the problem"), neither formalism is completely adequate for AI. There have been several extensions of these formalisms to time intervals. For example, dynamic logic was generalized to process logic (36,37). In process logic formulas are interpreted over paths, or sequences of discrete time points. In the version of Ref. 37 the two modal operators (in addition to the ones introduced by dynamic logic) are F ("first") and SUF (roughly, "until").  $S_1, \dots, S_n \models F p$  iff  $S_1 \models p$ .  $S_1, \dots, S_n \models \text{SUF } q$  iff, for some  $j$ ,  $S_i, \dots, S_n \models p$  for all  $i$ ,  $0 < i < j$ , and  $S_j, \dots, S_n \models q$ .

Interval temporal logic is a similar formalism, introduced by Moszkowski in conjunction with Halpern and Manna, which was applied to reasoning about digital devices (38). There are several other logics of time intervals, including a recent proposal by Halpern and Shoham (39). This logic, which extends point-based temporal logic in a way that is analogous to the way process logic generalizes dynamic logic, is one of few temporal logics in computer science (whether point-based or interval-based) that are not committed to the discrete view of time.

It was mentioned at the beginning that to date neither temporal logic nor dynamic logic have had much influence on AI. Some exceptions are found in Refs. 40–44.

## BIBLIOGRAPHY

1. J. M. McCarthy and P. J. Hayes, Some Philosophical Problems From the Standpoint of Artificial Intelligence, *Readings in Artificial Intelligence*, Tioga, Palo Alto, CA, pp. 431–450, 1981.
2. R. Fikes and N. J. Nilsson, "STRIPS: A new approach to application of theorem proving to problem solving," *Artif. Intell.* **2**, 189–208 (1971).
3. P. J. Hayes, The Naive Physics Manifesto, in J. R. Hobbs and R. C. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, 1984.
4. P. J. Hayes, Naive Physics 11: Ontology for Liquids, in J. R. Hobbs and R. C. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, 1984.
5. K. D. Forbus, Qualitative Process Theory, Ph.D. Thesis, Artificial Intelligence Laboratory, MIT, Cambridge, MA, 1984.
6. J. F. Allen, "Towards a general theory of action and time," *Artif. Intell.* **23**(2), 123–154 (July 1984).
7. A. Goldman, *A Theory of Human Action*, Princeton University Press, Princeton, NJ, 1970.
8. D. V. McDermott, "Planning and acting," *Cog. Sci.* **2**(2), 71–109 (1978).
9. D. V. McDermott, "A temporal logic for reasoning about processes and plans," *Cog. Sci.* **6**, 101–155 (1982).
10. D. V. McDermott, "Nonmonotonic logic II: Nonmonotonic modal theories," *JACM*, **29**(1), 33–57 (1982).
11. D. V. McDermott, Reasoning about Plans, in J. R. Hobbs and R. C. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, 1984.
12. J. de Kleer, Qualitative and Quantitative Reasoning in Classical Mechanics, Technical Report 352, MIT Artificial Intelligence Lab, Cambridge, MA, 1975.
13. R. Wilensky, *Planning and Understanding*, Addison-Wesley, Reading, MA, 1983.
14. R. Simmons, The Use of Qualitative and Quantitative Simulations, *Proceedings of the Third AAAI*, Washington, DC, 1983.
15. J. de Kleer and L. Seely Brown, A Qualitative Physics Based on Confluences, Technical Report, Xerox PARC Intelligent Systems Laboratory, January 1984.
16. B. Kuipers, "Commonsense reasoning about causality: Deriving behavior from structure," *Artif. Intell.* **24**(1), 169–203 (1984).
17. G. G. Hendrix, "Modeling simultaneous actions and continuous processes," *Artif. Intell.* **4**, 145–180 (1973).
18. C. Rieger, "An organization of knowledge for problem solving and language comprehension," *Artif. Intell.* **7** (1976).
19. T. Dean, Temporal Imagery: An Approach to Reasoning about Time for Planning and Problem Solving, Ph.D. Thesis, Yale University, Computer Science Department, 1986.
20. M. B. Vilain, A System for Reasoning about Time, *Proceedings of the Second AAAI*, Pittsburgh, PA, pp. 197–201, 1982.
21. J. L. Mackey, *The Cement of the Universe: a Study of Causation*, Oxford University Press, New York, 1974.
22. E. Sosa, *Causation and Conditionals*, Oxford University Press, New York, 1975.
23. D. Davidson, The Logical Form of Action Sentences, in N. Rescher (ed.), *The Logic of Decision and Action*, Pittsburgh University Press, Pittsburgh, PA, 1967.
24. B. F. Chellas, *Modal Logic*, Cambridge University Press, Cambridge, UK, 1980.
25. S. Kripke, "Semantical considerations on modal logic," *Acta Philos. Fenn.* **16**, 83–94 (1963).
26. A. N. Prior, *Past, Present and Future*, Clarendon Press, Oxford, 1967.
27. N. Rescher and A. Urquhart, *Temporal Logic*, Springer-Verlag, New York, 1971.
28. J. F. A. K. van Benthem, *The Logic of Time*, Reidel, Dordrecht, 1983.
29. A. Pnueli, A Temporal Logic of Programs, *Proceedings of the Eighteenth FOCS*, IEEE, pp. 46–57, October 1977.
30. A. Pnueli, "The temporal semantics of programs," *Theor. Comput. Sci.* **13**, 45–60 (1979).
31. E. A. Emerson and J. Y. Halpern, "Sometimes" and "Not Never" Revisited: on Branching versus Linear Time, *Proceedings of the Tenth ACM Symposium on Principles of Programming Languages*, pp. 127–140, 1983.
32. Z. Manna and A. Pnueli, Verification of Concurrent Programs: Temporal Proof Principles, Technical Report, Weizmann Institute, Department of Applied Mathematics, September 1981.
33. D. Gabbay, A. Pnueli, et al., On the Temporal Analysis of Fairness, *Proceedings of the Seventh ACM Symposium on Principles of Programming Languages*, pp. 163–173, 1980.
34. V. R. Pratt, Semantical Considerations on Floyd-Hoare Logic, *Proceedings of the Seventeenth FOCS*, IEEE, pp. 109–121, October 1976.
35. D. Harel, First-Order Dynamic Logic, in Goos and Hartmanis

- (eds.), *Lecture Notes in Computer Science*, Vol. 68, Springer-Verlag, Berlin, 1979.
36. V. R. Pratt, Process Logic, *Proceedings of the Sixth POPL*, ACM, pp. 93–100, January 1979.
  37. D. Harel, D. Kozen and R. Parikh, "Process logic: Expressiveness, decidability, completeness," *JCSS* 25(2), 145–180 (October 1982).
  38. B. C. Moszkowski, Reasoning about Digital Circuits, Ph.D. Thesis, Stanford University, Computer Science Department, July 1983.
  39. J. Y. Halpern and Y. Shoham, A Propositional Modal Logic of Time Intervals, *Logic in Computer Science*, Springer-Verlag, New York, June 1986.
  40. A. Fusaoka, H. Seki, and K. Takahashi, A Description and Reasoning of Plant Controllers in Temporal Logic, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 405–408, 1983.
  41. B. C. Moszkowski, Executing Temporal Logic Programs, Technical Report 71, University of Cambridge, Computer Laboratory, August 1985.
  42. Y. Shoham, Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence, Ph.D. Thesis, Yale University, Computer Science Department, 1986.
  43. E. Mays, A Modal Temporal Logic for Reasoning about Change, *Proceedings of the Annual Conference of the Association for Computational Linguistics*, Cambridge, MA, June 1983.
  44. M. P. Georgeff and A. L. Lansky, A Procedural Logic, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 516–523, 1985.

Y. SHOHAM AND D. V. McDERMOTT  
Yale University

## RECURSION

In this entry recursion is just a self-referential feature for procedures (and similar constructs) in certain programming languages. It was first available in LISP (qv) (1), the principal systems-implementation language for AI, and is available in most modern programming languages; for example, Pascal (2), Logo (qv) (3), Ada (4), and Modula 2 (5). Refs. 6 and 7 contain fine mathematical treatments of recursion's connections to mathematical recursive (or inductive) definitions, the subtleties of assigning formal semantics to recursion, and the use of proofs by mathematical induction to verify correctness of recursive procedures. Reference 8 is an excellent introduction to programming in LISP that effectively teaches one how to think recursively.

Essentially, recursion simply allows the instructions of a computer procedure to invoke the procedure itself. This brings up questions as to how it is possible to implement a thing like that and why anyone would want to use it. Suppose such a feature is allowed in computer procedures; the entry sketches how it is actually implemented. First, the usefulness of recursion is motivated by presentation of a simple, but illustrative example.

Some programming tasks create headaches for the programmer by ostensibly requiring that he or she devise methods to handle tedious "bookkeeping" subtasks that are subsidiary to the main tasks. Many times in such situations recursion can be used to free the programmer's mind of such unpleasant details; the details are handled instead by the computer implementation of recursion. Recursion in these cases enables the human programmer to produce succinct, conceptually clean

programs that are easy to understand and verify as being correct.

Sometimes in solving problems the intelligent thing to do, artificially or otherwise, is to systematically consider all the possibilities in a given situation. This can take the form of exhaustively and systematically searching some, perhaps complicated, structure (see Search). Two examples are searching a maze for desirable objects and searching a game tree (qv) for good moves. It is difficult in programming such searches to devise correct, clear strategies for the program to keep track of where it has already searched and where it still needs to go.

Here is an example. Imagine wanting to employ a programmable, electronic monkey to collect all the bananas in any "tree" of a certain type. A picture of the type of tree is presented in the tree-shaped diagram of Figure 1. This figure consists of dots and lines. The dots are called nodes. The bottom node in such a tree is called the root. For example, node 1 is the root of the tree pictured in Figure 1. Branching upward from each node are either two lines leading to two respective nodes or no lines leading to nodes. The entire tree is finite. The tree pictured in Figure 1 has 15 nodes that are numbered to facilitate some of the exposition below. Assume that any such tree has bananas only at its nodes. These assumptions as to the type of tree to be considered are merely to simplify the problem. The monkey is initially placed on the root of a tree and is capable of understanding and performing the following primitive tests and instructions that have obvious corresponding meanings. The tests are: "bananas.present.at.current.node" and "there.is.a.node.above." The instructions are: "pick.up.bananas.present.at.current.node," "climb.up.one.node.to.the.left," "climb.up.one.node.to.the.right," and "climb.down.one.node." Assume the monkey can be programmed with a block-structured language (as in the procedure "collect-bananas" given below) that allows recursion in procedures. The task is to program the monkey to gather all the bananas in any tree (of the type desired) if it is placed on the root of that tree. The monkey must also return to the root of the tree when it has finished gathering bananas. The problem is to make sure the monkey systematically traverses the tree without getting lost or missing any nodes.

Here is a crucial observation about the trees that makes a solution using recursion possible. Pick any node in one of the trees. That node together with the nodes and lines connected above it constitutes a tree itself. For example, if node 2 in the tree pictured in Figure 1 is selected, the connected substructure

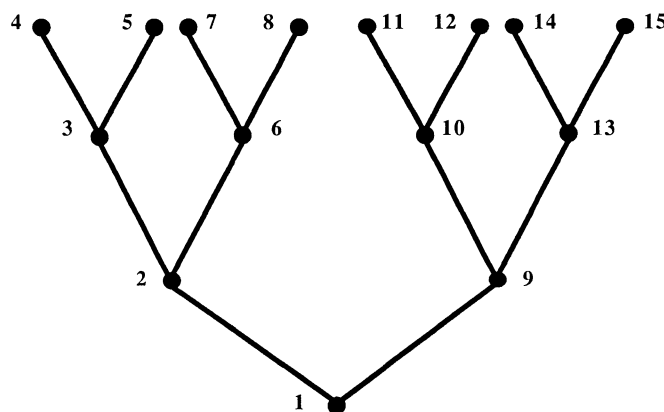


Figure 1

ture of the tree having nodes 2, 3, 4, 5, 6, 7, and 8 is itself another tree with root node 2. So is the substructure having nodes 6, 7, and 8 (the root is 6), as is the substructure consisting only of node 8 (the root is 8). Hence, from the point of view of the monkey, if it is sitting at node 6, it is ready to explore the subtree with nodes 6, 7, and 8. Below is a procedure called "collect.bananas" for accomplishing the programming task. The instruction "climb\_down\_one\_node," used twice, is to reposition the monkey after it has completed a subtree and ensure that the monkey finishes an invoking of "collect.bananas" at whatever node it was on when that invoking began.

```

procedure collect.bananas:
  begin
    if bananas_present.at.current_node
      then pick_up.bananas_present.at.current_node;
    if there_is_a_node.above
      then
        begin
          climb.up.one_node.to.the.left;
          collect.bananas;
          climb.down.one_node;
          climb.up.one_node.to.the.right;
          collect.bananas;
          climb.down.one_node
        end
      end
  end

```

This procedure is recursive because it invokes itself at two points inside the then- part of the second if-then. It is a problem for the monkey's computer to keep track of which invoking of the procedure it is currently working on and which it needs to get back to; with recursion this keeping-track is done by the computer, not the programmer. This computer bookkeeping of involkings prevents the programmer from having to devise an intricate bookkeeping algorithm for the task being programmed. Essentially, bookkeeping is handled once to implement recursion in the programming language, and then it does not have to be handled for other tasks.

If the sequence of involkings of "collect.bananas" is carefully traced for the example tree pictured in Figure 1, it will be seen that the instruction

```

if bananas_present.at.current_node
  then pick_up.bananas_present.at.current_node

```

is executed at the nodes of that tree in the numbered order of its nodes.

Recursion is nice for the programmer, but there is a cost of computational resources in its execution, i.e., the cost of the computer's bookkeeping to keep track of the involkings. On the one hand, for some tasks it is hard to see how to proceed without recursion. For example, it is difficult to produce and verify the correctness of an algorithmic solution to the famous Tower of Hanoi Puzzle (9) that does not employ recursion; however, the standard recursive solution is exquisite in its simplicity (9). On the other hand, the extra cost can be essentially eliminated in some commonly occurring special cases of recursion such as tail recursion. Tail recursion occurs when the recursive calls always occur at the end (or tail) of the procedure and can always be implemented with ordinary iterative loops. Reference 2 has a nice discussion of these cases and advice on when to use iteration rather than recursion.

A data structure called the stack is usually employed to implement the general cases of recursion for a programming language. The nitty-gritty details of how to do this for Algol,

e.g., may be found in Ref. 10. The general idea of how to do it by hand is sketched for simple cases; the computer implementations merely code an efficient, fleshed out version of the description.

To keep track by hand of recursive calls when executing a procedure such as "collect.bananas," it suffices to do the following:

1. Each time an invoking of the procedure recursively calls the procedure itself, put a blank sheet of paper on a stack of sheets on a table (the first sheet just goes directly on the table) and write on that sheet a succinct description of exactly where you were in the invoking of the procedure that made that self-call.
2. Invoke the self-call.
3. When any procedure call is finished, pull the sheet (if any) that is on the top of the stack and use that sheet to figure out what to do next in the execution of the procedure call that invoked the just-finished procedure call.
4. If an invoking finishes and there are no sheets on the stack, you are done.

The general case is more complicated. For example, some recursive procedures call other procedures that in turn call them; many recursive procedures have parameters whose values also have to be kept account of in the stack.

## BIBLIOGRAPHY

1. M. McCarthy et al., *Lisp 1.5 Programmers Manual*, MIT Press, Cambridge, MA, 1962.
2. N. Wirth, *Algorithms + Data Structures = Programs*, Prentice Hall, Englewood Cliffs, NJ, 1976.
3. B. Harvey, *Computer Science Logo Style*, Vol. 1, *Intermediate Programming*, MIT Press, Cambridge, MA, 1985.
4. R. Wiener and R. Sincovec, *Programming in Ada*, Wiley, New York, 1983.
5. J. Ogilvie, *Modula-2 Programming*, McGraw-Hill, New York, 1985.
6. H. Rogers, *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York, 1967.
7. Z. Manna, *Mathematical Theory of Computation*, McGraw-Hill, New York, 1974.
8. S. Shapiro, *Lisp: An Interactive Approach*, Computer Science Press, Rockville, MD, 1986.
9. D. Cooper and M. Clancy, *Oh! Pascal!*, 2nd ed., Norton, New York, 1985.
10. B. Randell and L. Russell, *Algol 60 Implementation: Translation and Use of Algol 60 Programs by Computers*, Academic Press, New York, 1964.

J. CASE  
SUNY at Buffalo

## REF-ARF

These are two programs written by Fikes in the late 1960s. REF is a nondeterministic programming language that is used to state problems for ARF to interpret. ARF combines constraint-satisfaction (qv) methods and heuristic-search (qv)

methods to solve problems. ARF is a problem solver (see Problem solving) similar in nature to the General Problem Solver (GPS) [see R. E. Fikes, "REF-ARF: A system for solving problems stated as procedures," *Artif. Intell.* 1, 27-120 (1970)].

J. ROSENBERG  
SUNY at Buffalo

## REGION-BASED SEGMENTATION

Image segmentation has often been treated as a problem by itself but is most important when used within a larger image-analysis system (see Image understanding). The brief description of the region-segmentation problem is either, given an image, to construct meaningful regions that correspond directly to objects or, given an image, to generate connected areas of the image that are uniform in some feature (color, depth, intensity, slope, texture, etc.). The desire is for the former; the result is usually the latter.

In a larger image-analysis system region segmentation plays an important role. A segmentation procedure operates on the input image and locates distinct regions. These regions serve as the basic units for further "higher-level" processing. The work by Nagao and Matsuyama (1) provides an example of the use of region segmentation within the context of a larger system.

There are several general categories of region-based segmentation techniques. These are first outlined, and then a more detailed description of each is given, illustrated by simulated segmentations on stylized images. Finally, a bibliography of relevant papers on this subject is included.

### General Segmentation Techniques

**Region Merging.** The image is divided into arbitrary elementary regions; then these are merged according to some criteria until no more can be merged. Early references used the term "region growing."

**Region Splitting.** Regions are split into two sets of regions based on the feature values of the points in the regions. Members of these sets may be further divided based on their feature values.

**Split and Merge.** A region is subdivided (in a regular, fixed pattern) if there are differences in the region. Then adjacent regions are merged if they are similar.

These general techniques are used in combinations, with knowledge about the image and for special-purpose segmentations, e.g., into two classes—the object and background. These three methods are illustrated with a highly stylized artificial image intended to show the flavor of the particular technique. Using real imagery might capture differences in performance of the methods but would fail to show all methods in the same light.

### Region Merging

Region merging is composed of three basic operations. The details of these operations differ from one implementation to the next, but all methods have these steps.

**Divide Image into Elementary Regions.** These regions may be arbitrary small regions (a  $5 \times 5$  block) as in Ref. 2 or connected components with exactly the same intensity value as in the early region-growing program of Brice and Fennema (3). The initial division will produce a large number of individual regions that will be combined into a much smaller number of regions. It is important that the image be subdivided as much as possible at this step since no more divisions will be made. Figure 1a illustrates a set of initial regions.

**Merge Similar, Adjacent Regions.** There are minor variations on this step, but the basic operation is to compare adjacent regions and to combine them if they are similar. Similarity can be measured over the entire region, along the boundaries of the two regions, or a combination of the two. Systems which depend on a strong model (2,4) can also consider possible interpretations of the region in the merging step. The simplest idea is to compare all adjacent regions and merge the most similar pair. Many variations are possible such as merging any that are very similar (a threshold on the similarity function), parallel merging rather than sequential, optimization of some global function, etc. All methods have the same general goals: reasonably compact regions that are generally uniform in some property. Figure 1b illustrates an intermediate step in the process.

**Final Step is Determining When to Stop.** The choice of the stopping criteria depends on the merging method, but it must consider the total number of regions, similarity of adjacent regions, shapes of region, knowledge of the scene, etc. Often a simple threshold on similarity is used or a maximum in the value of some global criteria.

**Discussion.** Basic region-growing techniques are simple to describe and analyze. The implementation is simplified by good data structures for handling the large number of initial regions, the determination of adjacent regions, computing similarities, and merging regions.

A number of parameters of thresholds must be selected. These may not be the same for all images. The choice of a threshold can greatly change the final segmentation by merging two different regions or failing to merge two pieces of one region.

### Region Splitting

Region-splitting techniques differ from region growing in the basic philosophy of their design. Region growing is based on

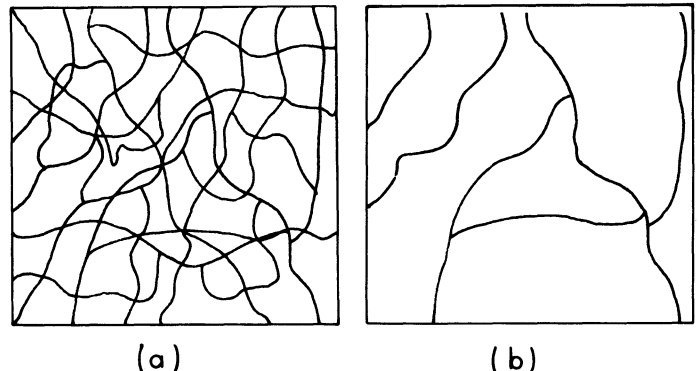


Figure 1. Region-growing example: (a) initial regions; (b) regions after several merging steps.

the assumption that adjacent regions that are very similar are part of the same object. Region splitting is based on the assumption that adjacent areas that are different are in different objects. Region-splitting techniques have concentrated on color images (multiple input parameters), and the additional color parameters help the performance. Split-and-merge methods discussed in the next section are similar. The distinction is in the method of splitting the regions into separate pieces.

The work of Ohlander (5) is the basis for much of the later work on general splitting techniques. Images are segmented by a simple recursive procedure. The basic step is to divide a region into two sets of regions. One set is composed of points within a given range of values for an image parameter; the other is composed of the remaining points. Each region in these two sets is then checked for a possible split. The work of Shafer (6) is basically similar in concept; it differs primarily in how the best split is chosen. Both of these methods use histograms of image features to determine the best threshold values. Figure 2 illustrates the operation of the procedure. Figure 2a shows two simulated color images for this scene.

### Region-Splitting Algorithm

Choose a region to subdivide. The initial region is one that covers the entire image; otherwise choose one from the current list of available regions.

Compute histograms of each image feature (e.g., intensity and color values). There will be one histogram for each image feature. If all histograms indicate uniformity (a single peak in each histogram), the region is completely segmented. Continue at the first step by choosing another region. Figure 2b shows the example histograms; both of these have two peaks, but the one on the left has two clearly separated peaks.

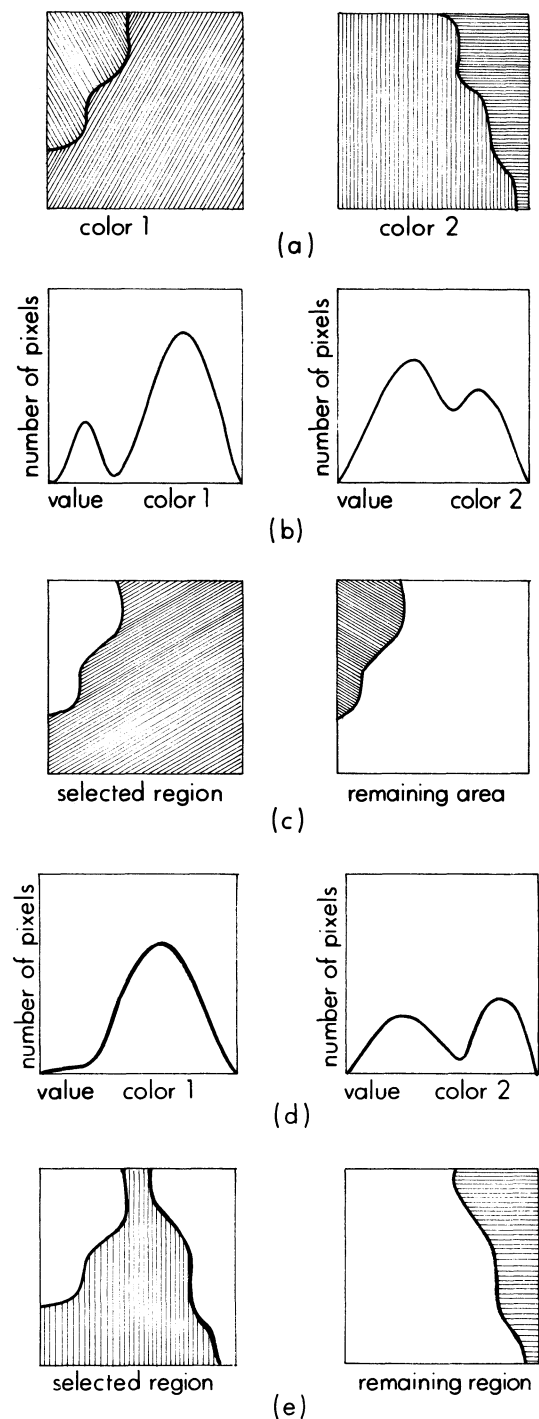
Choose a threshold range. Ohlander (5) made the choice based on the histogram. The best separated peak (a deep valley on both sides of the peak) is chosen with the two thresholds being the upper and lower boundary of the peak (adjacent minima, or inflection points). Shafer (6) chose all good thresholds and kept the best set of resulting regions. The larger peak in the histogram for color 1 (Fig. 2b) is chosen in the example.

Apply the threshold. This generates a set of connected regions. In the Shafer method several sets are generated, but only the one with the best regions is kept. The best regions are generally the most compact—fewest holes, smoothest boundary.

Extract the connected regions, and then extract the connected regions from the remaining points. Add these regions to the set of regions to be examined. At this step a control on the size of the allowable regions is possible. This gives the two regions shown in Figure 2c.

Continue at the first step by choosing another region.

**Discussion.** The two methods (5,6) are designed for segmenting color images (with multiple features) and require modifications to work as well for gray level images. They use the basic principle of removing regions that are obvious and easy first, then working on the harder regions. In the stylized example a threshold could be generated for either color, but the one for color 1 was better since there is less uncertainty about its location. The segmentation procedure is sequential.



**Figure 2.** Recursive region-splitting example: (a) simulated images; (b) histograms of simulated images; (c) regions from first step; (d) histograms of selected region; (e) regions selected on second step.

The power of this sequential extraction method compared to a global attempt to choose all optimum thresholds is discussed in Ref. 7. Multidimensional (two- or three-dimensional) histograms have also been used (8), but the advantages of the added complexity are not obvious when compared to a sequential selection of several thresholds.

The two methods differ in the procedure used to choose the best threshold range. Ohlander (5) tried to choose the best from the histogram alone. The same set of regions could cause a well-separated peak in several image features, and there is



no guarantee that the chosen peak gives the threshold values that produce the cleanest, most compact regions (small changes in thresholds primarily affect the boundaries). Shafer (6) eliminates the problem by choosing several thresholds (in different image features) and making the final decision based on the type of regions generated.

Histogram-based region-splitting methods are global in scope; regions are selected only if there is some indication in the global histogram. Adding more input parameters (colors) improves the results without complicating the procedure. The region-growing techniques can be more local, i.e., regions are joined based on their boundaries, not necessarily the features of the entire region and certainly not the features of the entire image. At each stage of processing the regions that have been selected are uniform in one feature (they may be divided later, but they represent reasonable partial segmentations).

### Split and Merge

The methods classified under split-and-merge techniques split the image in a uniform way and then merge adjacent similar subimages to form the final region structure. The work of Klinger and Dyer (9) and Tanimoto and Pavlidis (10) form the basis for the basic methods described here. The quad-tree data structure (see Reg. 11 for a detailed survey) is often used in conjunction with split-and-merge segmentation techniques. All of these segmentation algorithms are uniform in their application, i.e., the same procedure is applied many times to regions of increasing or decreasing size.

### Split-and-Merge-Algorithm

Start with the entire image. A very large image might be subdivided into "tiles" at the beginning without the uniformity check in the second step.

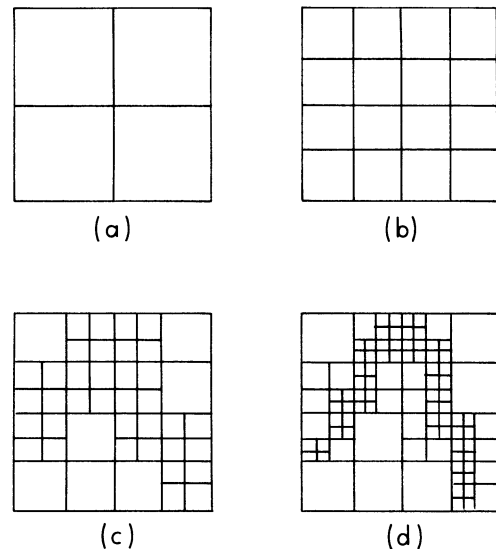
Determine if the image (or subimage on subsequent steps) is uniform. If it is uniform, the subimage is in only one region. If not, split the region into four equal regions (the quad in the quad-tree structure). See Figures 3a-d for a sequence of splitting steps.

Continue the division test for all new subimages. Stop at the single pixel level. The stopping criterion could be any size ( $2 \times 2$ ,  $4 \times 4$ ) depending on the desired resolution of the solution.

Merge adjacent similar subimages into a single region. This step is similar to the region-growing process except that the initial regions are computed in a different manner.

**Discussion.** Basic split-and-merge techniques are described straightforwardly and the implementation is primarily dependent on the data structure primitives available for constructing quad-trees. Like region growing, these algorithms must run to completion before reasonable results are available. Quad-tree data structures are fairly compact, but the structure for the segmentation procedure must store information about many regions rather than simply subdivisions.

The major variables in these methods are the criteria for uniformity of the regions, the merging similarity threshold, and the minimum subdivision size. This last variable makes such procedures ideal for multiple-level descriptions and enables segmentations of very large images in a uniform manner



**Figure 3.** Example of several steps of the splitting phase of the split-and-merge algorithm: (a) initial division; (b) second division, all positions are subdivided; (c) third level, several of the pieces are uniform and thus are not subdivided; (d) fourth level, the subwindows along the final border are divided again.

(12). This procedure lends itself to parallel implementation since, after the initial split, all decisions involve a small local area.

### Combinations and Other Methods

Most general region-segmentation techniques use some parts of one or more of these basic segmentation methods. One approach at combining methods is to combine edge and region information in 2-D histograms (edge-magnitude value and image-intensity value) (13). Alternatively, a threshold could be chosen and adjusted (up or down) to choose regions whose borders correspond to detected edges (14). Both of these methods were applied to special case (two-class) problems and proved difficult to extend to the general case.

The incorporation of scene knowledge in the segmentation is an important research area. This has been used heavily in region-growing systems (2,4) and in a more restrictive way in region-splitting systems. Histogram-based methods have been used extensively in problems where there are known to be a few (two or three) classes of objects (15,16). This is an explicit use of scene knowledge in the design of the algorithm. Kestner et al. (17) used initial regions—areas selected by an operator—and expanded the region to include all neighboring similar pixels. Nazif and Levine (18) developed an expert system for low-level segmentation that is driven by the data rather than by a global model.

Relaxation techniques are also used in segmentation, either as the primary method or in conjunction with other techniques (19,20). These techniques primarily aid in choosing from several possible alternatives. Nagin et al. (20) used a relaxation method to combine regions generated by the initial histogram-based thresholding scheme. This type of system proved to be difficult to use for the general case, and they have now incorporated many different ideas in a much larger, more general system (21) where segmentation into regions is a minor part of the entire operation.

## Evaluation

A variety of methods have been compared for application to specific domains, but general absolute conclusions about the best are not possible. For example, a study of segmentation algorithms for cytology (study of cells) (22) indicated that a simple histogram-based method was best overall. Other methods tended to be more sensitive to the choice of parameters and thresholds. These conclusions would not necessarily apply to all other domains. It is important to remember that segmentation into regions is not the final problem: It is the use to which the regions are put that is important. If the output of a segmentation program is sufficient for the problem being solved, that is a good segmentation. All too often, perfect segmentations are not even sufficient given the capabilities of the rest of the image-analysis system.

## Discussion

Early views of image analysis treated segmentation as a final problem by itself, but the more recent view is that segmentation is an integral part of the total solution and will not be done perfectly. This change in point of view has also been seen in edge-detection (qv) research.

All region-segmentation systems are trying to generate connected regions that are uniform in some property. They all have problems with regions that vary from bright to dark from one side to the other. Very thin regions usually cause problems; these are better suited to detection and description by edge- and line-analysis methods. All segmentation methods work well in some cases and poorly in others, but the basic methods discussed produce similar results with reasonable data.

Region segmentation will continue to be an integral part of image-understanding research with newer systems incorporating portions of some or all of the basic methods. The current directions are to incorporate more knowledge about the problem and to include more opportunities for feedback from the analysis programs to the region-segmentation system.

Further descriptions of techniques can be found in the references here or in several survey articles. Riseman and Arbib (23), Kanade (24), and Haralick and Shapiro (25) survey segmentation techniques over a period of years.

## BIBLIOGRAPHY

1. M. Nagao and T. Matsuyama, *A Structural Analysis of Complex Aerial Photographs*, Plenum, New York, 1980.
2. J. A. Feldman and Y. Yakimovsky, "Decision theory and artificial intelligence: I. A semantics based region analyzer," *Artif. Intell.* **5**, 349-371 (1974).
3. C. R. Brice and C. L. Fennema, "Scene analysis using regions," *Artif. Intell.* **1**, 205-261 (Fall 1970).
4. J. M. Tenebaum and H. G. Barrow, "Experiments in interpretation guided segmentation," *Artif. Intell.* **8**, 241-274 (1977).
5. R. Ohlander, K. Price, and R. Reddy, "Picture segmentation by a recursive region splitting method," *Comput. Graph. Im. Proc.* **8**, 313-333 (1978).
6. S. A. Shafer and T. Kanade, Recursive Region Segmentation by Analysis of Histograms, *IEEE International Conference on Acoustics, Speech and Signal Processing*, Paris, May 1982, pp. 1166-1171.
7. K. Price, "Image segmentation: A comment on "Studies in global and local histogram-guided relaxation algorithms," *IEEE Trans. Patt. Anal. Mach. Intell.* **6**(2) 247-249 (March 1984).
8. A. Hanson and R. E. Riseman, Segmentation of Natural Scenes, in A. Hanson and E. M. Riseman (eds.), *Computer Vision Systems*, Academic Press, New York, pp. 129-163, 1978.
9. A. Klinger and C. R. Dyer, "Experiments on picture representation using regular decomposition," *Comput. Graph. Im. Proc.* **5**, 68-105 (1976).
10. S. L. Tanimoto and T. Pavlidis, "A hierarchical data structure for picture processing," *Comput. Graph. Im. Proc.* **4**, 104-113 (1975).
11. H. Samet, "The quadtree and related hierarchical data structures," *ACM Comput. Surv.* **16**(2) 187-260 (June 1984).
12. J. D. Browning and S. L. Tanimoto, "Segmentation of pictures into regions with tile-by-tile method," *Patt. Recog.* **15**(1), 1-10 (1982).
13. D. L. Milgram and H. Herman, "Clustering edge values for threshold selection," *Comput. Graph. Im. Proc.* **10**(3), 272-280 (June 1979).
14. D. L. Milgram, "Region extraction using convergent evidence," *Comput. Graph. Im. Proc.* **11**(1), 1-12 (1979).
15. J. M. S. Prewitt, Object Enhancement and Extraction, in L. B. S. Lipkin and A. Rosenfeld (eds.), *Picture Processing in Psychopictorics*, Academic Press, New York, pp. 75-149, 1970.
16. D. L. Milgram and D. J. Kahl, "Recursive region extraction," *Comput. Graph. Im. Proc.* **9**(1), 82-88 (January 1979).
17. W. Kestner, M. Bohner, R. Scharf, and M. Sties, Object Guided Segmentation of Aerial Images, *Fifth International Conference on Pattern Recognition*, Miami, FL, December 1980, pp. 529-531.
18. A. M. Nazif and M. D. Levine, "Low level image segmentation: An expert system," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**(5), 555-577 (September 1984).
19. A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene Labeling by Relaxation Operations," *IEEE Trans. Syst. Man Cybernet.* **6**(6), 420-453 (June 1976).
20. P. A. Nagin, A. R. Hanson, and E. M. Riseman, "Studies in global and local histogram guided relaxation algorithms," *IEEE Trans. Patt. Anal. Mach. Intell.* **4**(3), 263-277 (May 1982).
21. G. Reynolds, N. Irwin, A. Hanson, and E. Riseman, Hierarchical Knowledge-Directed Object Extraction Using a Combined Region and Line Representation, *IEEE Workshop on Computer Vision, Representation and Control*, Annapolis, MD, April 30-May 2 1984, pp. 238-247.
22. S. Ranade and J. M. S. Prewitt, A Comparison of Some Segmentation Algorithms for Cytology, *Proceedings of the Fifth International Conference on Pattern Recognition*, Miami, FL, December 1980, pp. 561-564.
23. E. M. Riseman and M. A. Arbib, "Survey: Computational techniques in the visual segmentation of static scenes," *Comput. Graph. Im. Proc.* **6**(3), 221-276 (June 1977).
24. T. Kanade, "Region segmentation: Signal vs. semantics," *Comput. Graph. Im. Proc.* **13**, 279-297 (1980).
25. R. M. Haralick and L. G. Shapiro, "Survey: Image segmentation techniques," *Comput. Vis. Graph. Im. Proc.* **29**(1) 100-132 (January 1985).

K. PRICE  
University of Southern California

**RELATIONAL GRAPHS.** See Constraint satisfaction; Semantic networks.

## REPRESENTATION, ANALOGUE

A representation is a formal system that makes explicit how certain entities or types of information can be described. A key issue in AI is how to choose from the many possible representations the most appropriate one for a given entity. This is important because when knowledge is organized in a way that is unsuitable for the problem addressed, the solution may be very difficult or even impossible. One is familiar with this phenomenon through experiences where looking at a problem from one point of view is fruitless, yet the solution is readily apparent when one sees the problem from another viewpoint.

### Analogical vs Propositional Representations

It can be useful to divide representations into two classes—analogue and propositional (see Logic). This division corresponds to two theories of how humans represent information about the world. The following provides an extreme example of each class of representation. Suppose you were told that a walnut was in a bag, and the bag was in a box. If you were then asked if the walnut was in the box, you could easily give the correct answer. Some cognitive scientists would argue that you use an analogical representation to answer this question (1–3). That is, you “visualize” the walnut, bag, and box and then use this “image” to answer the question—just “look and see” if the walnut is in the box. Another possibility is that humans use a propositional representation (4,5). For example, the walnut problem could be represented as a set of propositions analogous to the following sentences: “The walnut is in the bag” and “The bag is in the box.” Then the question can be answered using a general inference method which includes the rule: If A is in B, and B is in C, Then A is in C.

In general, analogical representations have the following characteristics (6,7):

1. Each element of a represented situation appears once, and all its relations to other elements are simultaneously available.
2. A representation is amenable to apparently continuous transformation. For example, rotation of a mental image is analogous to the rotation of a physical object.
3. The structural relations between the parts of a representation are analogous to the perceived relations between the objects represented.
4. Analogical representations are manipulated by procedures that are often similar to physical or geometric simulation.

In contrast, propositional representations have the following properties (6,7).

1. Many different propositions may refer to a single element in a predicate-calculus-type representation (see Logic, predicate). Semantic networks (qv) may be used to represent propositions, in which case each element may appear as a node with its relations to other elements as links. There may or may not be a single node associated with each element.
2. Propositions generally represent discrete states rather than continuous change. Small successive increments of variables are necessary to represent continuous change.

3. The structure of propositions is not analogous to that of the situation. Propositions are either true or false.
4. Propositional representations are manipulated using the general rules of inference (qv).

### Which Type of Representations do Humans Use?

A debate in AI and cognitive science (qv) has centered around whether the internal representations used by humans are of an analogical or a propositional form. The debate has been polarized by some suggestions that not only is an analogical representation used but that an actual two-dimensional picture of a scene is formed in the head (1), whereas others have argued that the only internal representation needed is a propositional one (5). The importance of this analogical vs. propositional debate is diminished if you consider the fact that all computer implementations of representational systems are in some sense propositional, as they can be reduced to digital (discrete) symbol manipulation. This argument can be countered by pointing out that there are other levels at which the two types of representations do actually differ and that exploring these differences can lead to useful insights. In fact, there is evidence that both types of representation are useful for humans in particular and for AI in general (6,8). At the cognitive level processing is most naturally supported by some form of structural description that may be propositional (9,10), and while in the early stages of perceptual process, analog representations appear more suitable. For example, computational vision (qv) has demonstrated the usefulness of edge maps, intrinsic images, and other analogical representations in the first stages of visual processing. In general, when different computations are to be performed on the same entities, some of the computations would be simplified if the representation were analogous, and others would be best suited to a propositional representation. Thus, it may make sense to store the information using one type of representation, and then to transform it into the other type before performing certain computations.

The “imagery” vs. proposition question has clouded the real issues of knowledge representation. The types of relations that get encoded are of primary importance rather than the mode of representation. For example, Pentland has recently developed an interesting analogical representation of natural forms (11). In it a shape is described as a process-oriented representation that is analogous to how one would create a given shape by taking lumps of clay, forming them into particular simple shapes, and combining them to form the more complex shape. It is the relations here that are analogous, yet they could be encoded in a computer program that is basically propositional.

### Conclusion and Remaining Questions

The analogical vs. propositional representation debate has focused attention on some important questions for AI though the conclusion should be that the nature of the representation is of less importance than the question of what aspects or features of an entity, and relations between entities, should be encoded. The question of what to represent still remains, and it is here that the importance of analogical processes is paramount.

## BIBLIOGRAPHY

1. S. M. Kosslyn, *Image and Mind*, Harvard University Press, Cambridge, MA, 1980.
2. F. N. Shepard, "The mental image," *Am. Psychol.* **33**, 351-377 (February 1978).
3. D. L. Waltz and L. Boggess, Visual Analogue Representations for Natural Language Understanding *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, August 1979, pp. 926-934.
4. J. R. Anderson, *Cognitive Psychology and Its Implications*, W. H. Freeman San Francisco, 1980.
5. Z. W. Pylyshyn, *Computation and Cognition: Toward a Foundation for Cognitive Science*, MIT Press, Cambridge, MA, 1984.
6. P. N. Johnson-Laird, *Mental Models*, Harvard University Press, Cambridge, MA, 1983.
7. D. Ballard and C. Brown, *Computer Vision*, Prentice Hall, Englewood Cliffs, NJ, 1982.
8. D. A. Norman and D. E. Rumelhart, *Explorations in Cognition*, W. H. Freeman, San Francisco, 1975.
9. G. Hinton, "Imagery without arrays," commentary on S. M. Kosslyn, S. Pinker, G. E. Smith, and S. P. Shwartz, "On the demystification of mental imagery," *Behav. Brain Sci.* **2**, 555-556 (1979).
10. B. Kuipers, "Modeling spatial knowledge," *Cog. Sci.* **2**, 129-153 (1978).
11. A. P. Pentland, Perceptual Organization and the Representation of Natural Form, SRI Technical Note 357, SRI International, Menlo Park, CA, 1985.

D. WALTERS  
SUNY at Buffalo

## REPRESENTATION, KNOWLEDGE

Computer solutions to many problems in AI depend more on the availability of a large amount of knowledge than on sophisticated algorithms. That is, although certain games can be played simply by searching through a large subset of the board positions that can be reached from a given position, performing a medical diagnosis depends on the knowledge of the kinds of diseases, their symptoms, and their interactions with the human body. Similarly, it has been argued that for a program to communicate in a natural language, it is necessary that it in some sense know about the world. Knowledge representation is the study of ways of supplying programs with such knowledge. The term "knowledge base" is used to refer to the body of knowledge made available to the program.

First, it must be pointed out that a database-management system is not an adequate solution to the knowledge-representation problem. Among other things, such systems cannot represent and reason with such facts as "All persons are mortal." Combined with the fact "Socrates is a person," it is clear that the knowledge base should be able to answer the question "Is Socrates mortal?" The answer to that question is an example of implicit knowledge. Second, although there is agreement that there is a knowledge-representation problem, there is no agreement on how the problem should be solved or what in fact the solution should offer. This variety can be seen in the responses to a questionnaire sent to various workers in the field (1). More recently, a variety of approaches to the problem are presented in Ref. 2. Other surveys of the field of knowledge

representation are those by Barr and Davidson (3) and Mylopoulos and Levesque (4).

This entry begins with a short discussion of the terms used to describe things that are stored in a knowledge base. This is followed by a brief history of the field. This history introduces many concepts that are more fully described later. The main body of the entry discusses the issues that must be addressed by someone choosing or designing a knowledge-representation scheme. First the choice of notation and operations on knowledge bases is discussed. Examples of notations are logical languages, graph structures called semantic networks, and some programming languages. Knowledge-base operations include storing and retrieving facts and deriving implicit knowledge. The second issue is the specification of a semantics (qv), a formal way of assigning meaning to facts. The third issue involves the choice between procedural and declarative schemes. The fourth issue is the choice of mechanisms for organizing knowledge. This includes concepts such as specialization hierarchies and the grouping of facts into structures called frames (see Frame theory). The last issue involves the choice of an inference (qv) scheme, a mechanism for retrieving implicit knowledge. Subsequently, some examples of knowledge-representation schemes are presented. These include systems such as PROLOG, FRL, and KRL. Finally, some areas of active research are mentioned.

**Vocabulary.** A knowledge base is a collection of facts about the world. The term "object" is used to refer to a data structure in the knowledge base that is intended to denote an entity in the world. For example, a knowledge base might contain an object representing the person Socrates. Objects are also used to represent abstract concepts such as the class of all people. Terms that are often used to mean the same thing as the word object are "concept," "constant symbol," and "unit."

A fact is a statement that some relationships hold or are believed to hold between the entities denoted by some objects. For example, one might state that Socrates' birthplace was Greece. These statements about the world are variously called sentences, clauses, formulas, assertions, or properties of objects depending on the particular knowledge representation.

**History.** Philosophy has long concerned itself with the study of knowledge. Most of this work, although it may very well be relevant to knowledge representation, has not in fact strongly influenced the field. However, mathematical logic (qv) was one of the first formalisms that was proposed as a knowledge representation (5) and is still the basis for a large part of research in the field. Some important issues arising from the use of logic are discussed in Ref. 6.

A book edited by Minsky (7) contains one of the first strong arguments for the necessity of knowledge representation for AI. This book contains a collection of papers describing programs whose performance depends on a knowledge base. In particular, many of these papers argue that an organized body of knowledge is necessary in order that the programs might find solutions to problems in a reasonable amount of time. One of the most important papers in this collection is Ref. 8, where Quillian introduces the notion of semantic network (qv). A semantic network is a directed labeled graph in which the nodes represent objects and the arcs express relationships between these objects. This way of representing knowledge has been incorporated into many subsequent systems.

Cognitive science (qv), the study of mechanisms of intelli-

gence, has been a major source of ideas for knowledge representation. Quillian's semantic nets were in fact proposed as a mechanism for accounting for some of the associative features of human memory. A second popular scheme for representing knowledge, the production system, was also introduced as such a mechanism (9). In a production system knowledge is expressed as a set of rules, each of which specifies a set of actions to take should certain conditions hold (see Rule-based systems). For example, a rule might say "if there is a block on top of the block that is to be moved, move the top block." This scheme is the basic representation of knowledge used by most existing expert systems (qv) (10). A large collection of papers describing production systems is Ref. 11.

The straightforward application of mathematical logic to representation encountered several difficulties. Most important is the fact that unrestrained attempts to derive implicit knowledge are computationally intractable. A second problem, called the frame problem, involves the need to specify all facts that do not change as well as those that do change when describing activities. As a reaction to these problems, a number of procedural formalisms were introduced. Among these were PLANNER (qv) (12) and its successor, CONNIVER (qv) (13). These languages mix the use of the standard programming language LISP (qv) with the use of new control structures where procedures can be invoked through patterns as well as by name.

As semantic-network schemes proliferated, it became clear that there was no agreement as to what a semantic network was or how nodes and arcs were to be interpreted. In response, Woods (14) suggested that what was needed was a well-founded semantics similar to that of some logics. This paper spawned a new generation of semantic-network formalisms, many of which are described in Ref. 15. Among these were schemes that used semantic networks to represent sentences of mathematical logic.

At about the same time Minsky (16) introduced frames as a way of representing knowledge. A frame is a data structure containing an aggregation of facts about an object. This paper too has had a significant influence on subsequent approaches to representation. Examples of the resulting frame-based representations are KRL (qv) (17), FRL (qv) (18), and PSN (19).

Most recently there has been an attempt to integrate different approaches. An example of this approach is KRYPTON (qv) (20) in which there are two components: a frame-based component in which terms are defined and a logic-based component in which facts about the world are stated. For example, a term might be "bachelor," which is defined as an unmarried male person, whereas an assertion might be used to represent the fact that John is a bachelor.

**Knowledge Base Operations.** The specification of a knowledge representation consists of two major components: a description of the notation used to express facts and a description of the operations that can be performed on a knowledge base. These components are discussed in this section.

A notation in which facts can be expressed is essential to any knowledge representation. One example of such a notation is some version of logic in which a fact might be

$$(\forall x)(\text{Person}(x) \supset \text{Mortal}(x))$$

A notation includes some notion of a syntax that describes and decides when a sentence is well-formed. In addition to the

syntax specified by the notation, some knowledge representations allow facts to be further constrained by other facts, often called schemas, in the knowledge base. In its simplest form such constraining facts are little more than specification of abstract data types analogous to those of programming languages or data models. For example, one might enter a schema for statements about people that constrains their ages to be numbers. Any attempt to state that the age of a person is something other than a number would then be invalid. The term "metaknowledge" (see Metaknowledge, metarules, and metareasoning) is used to describe statements about other statements. In addition to constraining the syntax of an expression as in the above example, metaknowledge can constrain the semantics of a statement as in " 'Jack is 5 years old' is true," or its use, as in "When determining a person's birth date, use his age." Further uses of metaknowledge are the subjects of much current research.

The most basic function provided by a knowledge base is to store facts. This might involve checking that a fact is well-formed, checking that it does not conflict with previously stored facts, altering the form of the new fact, and/or storing some additional knowledge that follows from the new fact. Once the knowledge is stored, it must be possible to retrieve it. A program might query a knowledge base in two different ways: The first asks a yes-no question such as "Is Socrates a person?" and the second is a retrieval question such as "Return a list of all persons."

The third function performed by a knowledge representation is inference (qv), i.e., the retrieval of implicit knowledge. This function would answer the question "Is Socrates mortal?" from the knowledge that "All persons are mortal" and "Socrates is a person." In the past, knowledge-representation schemes have been proposed without functions for deriving implicit knowledge. Inference was therefore part of the job of the particular problem-solving program. On the opposite end of the spectrum, given a knowledge representation with a powerful enough inference scheme, one can develop a solution to a problem simply by storing the necessary facts in a knowledge base. For example, one could create a body of knowledge about diseases and their symptoms, add the list of symptoms for a particular patient, and ask the system to infer the disease, if any, afflicting that patient.

Inference, in general, can involve the use of a large amount of computer time. In particular, the line between the retrieval of implicit knowledge and the solution of a problem is not clear. In some representations one in fact need only specify the relevant knowledge, and the inference engine will solve the problem. Other representations require extra facts that specify how a particular problem is to be solved. For example, one might state that "When the patient has chest pains, try to infer heart disease before other diseases." Such facts are examples of control knowledge. Without control knowledge, the more powerful schemes are generally computationally intractable: It has therefore been argued that weaker inference schemes are desirable (21).

Other common operations on knowledge bases include functions that return the objects mentioned in the knowledge base, return the definitions of concepts, and remove facts from the knowledge base.

**Semantics.** Suppose a knowledge representation allows one to write the statement "mortal (Socrates)." One probably wants this to mean that the person named Socrates is mortal.

The meaning of objects and statements—i.e., the relationship between objects in the representation and entities in the world—is a part of the semantics (qv) of a knowledge representation. The meaning of a statement is usually taken to be a truth value;  $T$  if the statement is true,  $F$  if the statement is false. A popular way of formally assigning a semantics to logical languages is due to Tarski (22). An interpretation is a mapping of objects in the knowledge base to elements of some set  $D$ , and a mapping of  $n$ -ary relations in the knowledge base to subsets of the set of  $n$ -tuples over  $D$ . A simple knowledge base sentence is one that states that a relation holds between some objects. Such a sentence is true with respect to an interpretation when the  $n$ -tuple formed from the interpretation of the objects is an element of the set corresponding to the relation. One can then define the truth values of sentences as the composition of the truth values of their parts. An interpretation is a model of the knowledge base if all sentences in the knowledge base are true in that interpretation.

Given a way of interpreting facts, there are several questions one can answer about a knowledge base. First, if there is no model for a knowledge base, the knowledge base is inconsistent. This results when contradictory facts can be derived. A simple example of an inconsistent knowledge base is one containing both "Socrates is a philosopher" and "Socrates is not a philosopher." Second, one can prove that an inference scheme derives only sentences true with respect to the model. This is called inferential validity or soundness. Third, one can ask if a knowledge base is complete, i.e., whether every true statement expressible in the language can be derived from the knowledge base (see Completeness).

### Procedural and Declarative Representations

One of the major controversies in AI in the 1970s was the procedural vs. declarative debate in knowledge representation. A declarative representation is, e.g., a version of logic where the knowledge base might contain the facts

$\forall x \text{ person}(x) \supset \text{mortal}(x)$   
 $\forall x \text{ dog}(x) \supset \text{mortal}(x)$   
 $\text{person}(\text{Socrates})$   
 $\text{person}(\text{Helen})$

This states that all persons are mortal, all dogs are mortal, Socrates is a person, and Helen is a person. Given an inference rule such as *modus ponens*, one can find out whether Socrates is mortal.

A procedural representation, on the other hand, is one in which facts are represented in terms of programs and data structures. The knowledge base above might be expressed as

```
procedure person (x)
  if (x = Socrates) or (x = Helen) then return true
  else return false

procedure mortal (x)
  if person (x) then return true
  else if dog (x) then return true
  else return false
```

Here one could ask whether Socrates is a person by executing

person (Socrates)

and whether Socrates is mortal by executing

mortal (Socrates)

The declarative approach has advantages of flexibility and modularity. First, most implementations of the inference rule for declarative representations allow queries such as

mortal ( $x$ )

where  $x$  is a variable. The query would return the result yes and in addition provide a value such as Socrates for the variable. This facility is possible in the procedural representation but requires that each procedure be explicitly programmed to check for variables and return appropriate values. Second, adding the fact that horses are mortal,

$\forall x \text{ horse}(x) \supset \text{mortal}(x)$

to the declarative knowledge base is quite simple. In the procedural case one would have to modify an existing procedure to make this change.

The advantage of the procedural representation lies in the control of search. In the above simple example the procedure mortal ( $x$ ) first checks whether  $x$  is a person and then whether  $x$  is a dog. This choice of algorithm might have been made with the knowledge that the mortality of people is queried more often than that of dogs; hence in general the inference is faster if "person" is called first and checking whether  $x$  is a dog is avoided entirely. In a declarative knowledge base, on the other hand, search is required to find the statements that lead to the conclusion.

In order to overcome the disadvantages of procedural languages, in particular the limited capability for interaction and the difficulty of adding new knowledge, languages have been designed that have some of the features of declarative languages. This is done by changing the method for invoking procedures: instead of writing  $\text{man}(x)$  to invoke a procedure called "man," one writes code such as  $\text{goal}(\text{man}(x))$ , which means "run a procedure whose description states that it might be able to find out if  $x$  is a man." This is known as pattern-directed invocation. Such systems generally allow more than one procedure to satisfy the pattern. If one procedure is unable to provide an answer (it fails), the others are tried. This is called backtracking (qv). However, by incorporating pattern-directed invocation and backtracking into a system, one has again given up some control of search.

Instead of, or in addition to, augmenting a procedural language with pattern-directed invocation, authors have proposed knowledge representations that contain both a declarative component and a procedural component. Some inference is done by the inference rule defined over the declarative structures, but under certain conditions, inference is performed by procedures contained within the knowledge base. For example, in some knowledge representations one might specify that the successor property of numbers should be computed by the LISP function ADD 1 rather than through inference.



## Organization of Knowledge

Much of the work in knowledge representations has focused on proposals for organizing the knowledge in knowledge bases. This was largely motivated by a desire to make retrieval and inference as efficient as possible. However, it is important to note that the organization for efficient retrieval need not be visible at the level of the notation (23). An implementation of a logical language, e.g., could use any number of indices to relate facts. On the other hand, if the representation is proposed as a mechanism by which the human brain actually stores knowledge, the retrieval aspect is significant.

Another motivation for the organization of knowledge might be called conceptual efficiency. Since knowledge bases will need to contain a large number of facts, mechanisms must be present that aid in the understanding and maintenance of these bodies of knowledge. The situation is similar to that which leads to the introduction of structured programming and data abstraction as tools for building large computer programs.

**Networks.** A significant number of the knowledge representations proposed in the literature are called semantic networks (qv). What such representations have in common is that knowledge is represented by a labeled, directed graph whose nodes represent concepts and/or objects in the world and whose arcs represent relationships between these objects and concepts. For example, a knowledge base might consist of nodes labeled John, person, and 23 and contain an arc labeled "an instance of" from John to person and a second arc labeled "age" from John to 23. Such a graph is intended to represent the fact that John is a person and that his age is 23. This structure clearly makes certain retrievals efficient: All of the facts about John are directly accessible from the node labeled John. Furthermore, semantic networks have an obvious graphic notation that aids a human in understanding the contents of the knowledge base.

Semantic networks have often been proposed as a way of storing sentences from logical languages (e.g., see Ref. 24). In these cases the intention is purely to improve the speed of access to facts. Similarly, the original proposal by Quillian for a semantic network was intended as a model of the way humans store knowledge, again with an emphasis on the speed of access.

**Frames.** Minsky's proposal that knowledge be represented by a set of frames sparked a new generation of knowledge representations (16) (see Frame theory). A frame is a complex data structure representing a prototypical situation. For example,

```
frame PERSON
  father: PERSON
  eye-color: COLOR default = blue
  age: NUMBER constraint age < father.age
```

represents the concept PERSON. The contents of the frame is a list of slots that define relationships to other frames that play various roles in the definition. Thus, the definition of the slot father states that an object described by the frame PERSON plays the father role with respect to any person. In addition, a slot can contain a default that is used in the absence of

other information. In the example, unless told otherwise, the system will infer that a person has blue eyes. Another common mechanism associated with slots is the ability to constrain possible fillers with respect to the fillers for other slots. In the example a person's age must be less than the age of his/her father.

Other things that may be associated with frames are programs that describe how to use them, similarity links that link a frame to a frame and are used when a situation is encountered that does not fit the original frame, and exceptions that describe how situations might fail to match a frame. The term "procedural attachment" is used to describe the mechanism in which programs are associated with objects or facts. Minsky proposed that procedures could be attached to slots as well as frames. Procedures could be supplied to compute the value of a slot, store a value into a slot, and check that a situation is described by the frame. For example, one might have defined PERSON as follows:

```
frame PERSON
  birth-year: YEAR
  father: PERSON
  eye-color: COLOR default = blue
  age: NUMBER procedure = COMPUTE-AGE

procedure COMPUTE-AGE ( )
  return (CURRENT-YEAR ( ) - (current-person.birth-year))
```

Here the value of the age slot is computed by a procedure called COMPUTE-AGE, which subtracts the person's birth year from the current year.

In the various frame-based representations, some of which are described below, a variety of names are used for frames. These include the words "concept," "class," and "unit."

**Organizational Relationships.** In semantic-network and frame-based knowledge bases it is common to distinguish between objects representing individuals such as Socrates and objects representing classes or concepts such as person. When referring to knowledge base objects, the former are called instances, tokens, or individual concepts and the latter are called types or classes. When this distinction is made, there are several relationships that have been found useful in organizing knowledge.

Instance-of is the relation that relates a token to its type or an instance to a class. This relationship classifies the objects of the knowledge base. For example, some objects in a knowledge base might be of type "person" while others are of type "dog." Classification can be used

1. to enforce knowledge base consistency, e.g., if there is a mechanism for stating that the filler for a slot age in a frame such as person must be of a given type, maybe number, an attempt to state that the age of John is "Smith" would fail and
2. to make search more efficient: If the system is looking for an object whose student number is 98765432 and it is known that only students have students numbers, then only the set of instances of "student" must be searched.

Some languages (e.g., PSN) allow classification to be recursive; i.e., classes can or must themselves be instances of

classes. In this case the class person might be an instance of the class of all classes describing animate objects.

Specialization occurs when one class is defined as a more refined version of another. For example, "student" might be a specialization of "person." The specialization relationship is often called IS-A. One benefit of specialization is that it allows information that applies to a number of classes to be specified only once at a more general level. Thus, when properties of people are specified for the class person, and student is made a subclass of that class, it is only necessary to specify for "student" those things that are not inherited from "person."

Aggregation refers to the relationship between a frame and its slots or an object and its components. Thus, a person can be viewed as the aggregation of a set of body parts: arms, legs, head, etc. In general, there is more than one way of viewing an object as a set of components. A person might also be viewed as a social object consisting of the aggregation of a name, an address, an occupation, etc. Components of an object might themselves be decomposed. For example, the address of a person might have the parts street number, street name, city, etc.

A fourth way of organizing objects in a knowledge base is through partitions (25) or contexts. A partition is simply a subset of the objects in the knowledge base. One use of a partition is to group together all of the facts about a given situation, e.g., the case history of a particular patient. Here partitioning is a useful mechanism for limiting search for relevant facts. Alternative uses for partitions are the representation of different possible worlds or the representation of the facts believed about the world by different agents known to the system. For example, the system might need to represent the fact that when John is reasoning, he believes that the sky is red. In this case, partitions are often called belief contexts. Partitions are usually organized into a hierarchy by a relationship that could be called "formed from." One specifies how a partition *B* is formed from a partition *A* by stating what objects and relationships have been added to or deleted from *A* to obtain *B*. An object that belongs to a partition is said to be visible in that partition. In particular, all objects in *A* that were not removed in forming *B* will be visible in *B*.

**Inference.** Given a choice of data structure, it remains to specify how implicit knowledge is derived from a knowledge base. In a purely declarative representation one states how a true derived statement is related to the facts in the knowledge base; an algorithm for testing this relationship is left to a particular implementation. For a procedural formalism one supplies control structures that interpret the procedures that express facts. In either case, inference schemes usually depend on some sort of generalized matching.

**Matching.** Matching involves the comparison of a pattern with an object in the knowledge base. The match succeeds if the object is unstructured and the pattern primitively matches the object or if the object is structured and each part of the pattern successfully matches one or more parts of the object. For example, if objects are either letters or lists of letters, the pattern (*A B C*) would match the object (*A B C*).

Pattern matching would be uninteresting if a pattern only matched itself. What makes it powerful are various ways of specifying a range of objects that can match the pattern. In the above example one might allow a pattern element that matches any letter. For example, (*A ? C*) could be a pattern that matches any list of three letters of which the first is an *A*

and the last a *C*. Another common mechanism is a notation that matches more than one element. The pattern (*A \* C*), for example, might match a list starting with an *A*, ending with a *C*, and having any number of intermediate elements. A large variety of mechanisms are possible, and the choice depends on the representation language. There is also a trade-off between the complexity of the pattern and the computational efficiency of the pattern matcher.

A third important ingredient of pattern matching is the ability to associate variables with pattern elements. When a match succeeds, the values of these variables are the parts of the object that matched the corresponding pattern elements. For example, the pattern (*A \* x C*) might match any list beginning with *A*, ending with *C*, and having any number of elements in between, and that when the match succeeds, the value of *x* will be the list of elements in between. Thus, after matching (*A W X Y C*), the value of *x* would be (*W X Y*).

When both the pattern and the object may contain variables corresponding to pattern elements that can match any object element, the resulting operation is a form of unification. In effect, there is no distinction between pattern and object. This was introduced as part of the resolution method for proving theorems in logic (26) (see Theorem proving). In this case both the pattern and object can be viewed as general expressions including variables. The result of unification is the most general substitution of variables that makes the pattern and object identical. Unification has in the past been used in matching linear expressions; recent work has extended unification to frame-based structures (27).

The above discussion has focused on a kind of matching where the parts of a structured pattern are put into correspondence with the parts of an object having the same structure. This is known as syntactic matching. Semantic matching, on the other hand, involves finding a correspondence between the pattern and object based on some description of the function or role of the parts in an object. For example, a pattern might specify that a successful match would be a set of two people, one of whom is the husband in a marriage and the other the wife in the same marriage; this might be written as (*x:(a husband), y:(a wife)*).

Various knowledge representations make different uses of matching. First, in pattern-directed invocation a pattern associated with a procedure is matched with the current object of interest to see whether that procedure is relevant. Second, unification plays a basic role in the inference mechanism of resolution-based systems. A third use is classification, in which the type of an unknown input is determined by the pattern that it matches. A fourth use is confirmation, where an object is checked against a pattern to verify that it is the desired object. A fifth use is decomposition. Finally, the nature of the failure of an attempt to match a pattern can be used to guide the correction of some error.

**Declarative Inference.** The main example of inference (qv) in a declarative knowledge base is the provability relation of first-order logic. This consists of a set of syntactic transformations of true sentences that will result in more true statements. For example, *modus ponens* states that if one has a sentence of the form "*A implies B*" and another of the form "*A*," the sentence "*B*" can be proved.

A second example is inference through spreading activation as introduced by Quillian. This is also called marker passing. The idea here is that a problem can be solved by locating

objects that are at the same distance from two or more starting points in a network. The distance between nodes is defined as the length of the shortest path between the nodes. This does not necessarily correspond to the path with the fewest connections because arcs may be assigned individual lengths. When the arcs are all assigned the same length, an algorithm for spreading activation involves marking each of the starting nodes with a different color and then successively marking all nodes connected by an arc to a colored node with the same color. When a node is colored with all of the colors, it is the solution to the inference. For example, inference starting at the nodes elephant and gray in a network might be the same distance from the concept Clyde, a gray elephant. Fahlman's NETL (28) is a knowledge representation based on such a scheme. Two important issues concerning marker passing mechanisms are the definition of distance between nodes in a graph and how various arcs restrain the passing of markers (colors).

A third form of inference involves testing the truth of the subsumption relationship in frame-based knowledge bases. One frame is said to subsume another if all instances of the second are instances of the first. Given a test for subsumption, one can infer, e.g., that a male person who has never married is a bachelor.

**Control Structures.** When a procedural representation makes use of pattern-directed invocation, it is often the case that more than one procedure matches the pattern. One aspect of the control structure determines how such conflicts are resolved. A second aspect of the control structure involves the direction of inference, i.e., the mechanism that chooses the patterns used to select procedures.

In the past, the mechanism for conflict resolution has usually been wired into the inference engine. This mechanism incorporated such rules as use the procedure appearing first in the knowledge base or use the procedure with the most complicated pattern. More recently there has been research into the use of an inference engine using a knowledge base of control knowledge to resolve conflicts (29,30). This has the advantage that new knowledge about resolving conflicts can be added without the need to recode the inference engine.

Forward-chaining inference arises when procedures are invoked if their patterns match facts about to be added to or removed from the knowledge base. Generally, such procedures will add and remove facts of their own thus triggering yet other forwardly chained inferences. Backward chaining inference works in the reverse direction. Trying to match a given pattern against the knowledge base may trigger procedures which set other pattern matching goals in an attempt to achieve the original pattern match (see Processing, bottom-up and top-down).

## Examples

**PROLOG.** PROLOG (qv) is a programming language based on a subset of first-order logic (31). The interpretation of a set of sentences in logic as a program is facilitated through some conventions and extra syntax that allow the activity of the theorem prover to be controlled. Since a language based on logic can be used to represent knowledge, PROLOG is often called a knowledge-representation language (see Logic programming).

A PROLOG knowledge base (or program) consists of a set of

horn clauses. A horn clause is simply a statement of the form "if condition 1 and condition 2 and . . . then conclusion." The conclusion is called the consequent, and the conditions are called antecedents. The conditions and the conclusion are each an expression of a relationship between terms that are names of objects, variables, or functions of terms. A PROLOG clause expressing the fact that all men are mortal might be written as

$$\text{mortal}(X) \leftarrow \text{man}(X)$$

Note that the consequent is written on the left side of the arrow.

Inference in PROLOG can use any theorem prover for first-order logic. Actual implementations, however, make use of the restricted version of logic in order to make execution relatively efficient. In addition, a backward-chaining theorem prover with automatic backtracking is used. Inference is controlled by the order in which clauses are stored in the knowledge base and by control operators included within the clause. The rules used by the inference scheme are the following:

Clauses are tried in the order in which they are stored in the knowledge base.

The processor tries the antecedents of a clause in the order in which they are stored.

Search is depth-first (see Search, depth-first), i.e., the antecedents of the most recently introduced clause are considered first.

Special syntax is provided to restrict search. One example of this is the cut operator, usually shown as an exclamation point (!). This operator indicates that if search fails for any antecedent following the cut in the clause, the goal that the clause is being used to prove fails immediately. This is used in situations where it is known that if an approach fails after a certain point, all other attempts will fail.

**PLANNER.** PLANNER (qv) is a procedural language based on pattern-directed invocation. A system written in PLANNER consists of a database of assertions; these are simply LISP data structures. For example, one might represent the fact that Socrates is a man with the assertion (MAN SOCRA- TES). In addition to the database of assertions one can define a set of procedures called theorems or demons (qv). A theorem is executed when its pattern matches the argument of a database operation. There are three kinds of theorems.

Antecedent theorems are invoked when their patterns match structures that are being added to the database. For example, one might write an antecedent theorem that asserts the fact (MORTAL X) whenever the fact (MAN X) is asserted. Since this could trigger the execution of other antecedent theorems, a form of forward-chaining reasoning occurs. Similarly, erasing theorems are executed when their patterns match facts that are to be removed from the database.

Consequent theorems are invoked in response to a goal statement. A goal statement is used to retrieve information from the database; when the pattern of a goal matches the pattern of a consequent theorem, the theorem is used to derive implicit knowledge matching the goal pattern. The body of the theorem can assert or remove facts or try to prove other goals. For example, a consequent theorem for solving the goal (MORTAL X) might try to prove the goal (MAN X). The theorem can choose to fail if it cannot show that the pattern is true. Since

more than one theorem might match a goal pattern, if one fails, the system automatically tries another. Note that when this backtracking occurs, it is necessary to undo the effect of all changes to the database resulting from the body of a theorem to the point of its failure.

**Production Systems.** A large number of expert systems use some form of production system as the representation language. The basic features of such a representation are a global database containing the results of a computation and a set of production rules that operate on the database. A production rule consists of a procedure called the body of the rule and an associated pattern. Inference consists of a cycle in which a rule is found whose pattern matches the database, and then its body is executed. The body of a rule will generally make some changes to the database by adding and deleting structures. Since rules cannot invoke other rules directly, all communication is done through the database. The rules are therefore said to be loosely coupled.

There are many sources of variety in production systems. First, the size and complexity of the global database can vary. The production system proposal by Newell and Simon restricted the contents of the database to a small number of lists. In this work the database was viewed as a model of human short-term memory, and all knowledge stored in long-term memory was to be represented as rules. HEARSAY-II (qv) (32), on the other hand, uses a global database called a blackboard in which a large number of frames partitioned into layers are stored. Second, production systems can vary in the complexity of the pattern allowed; sometimes the pattern may contain calls to arbitrary functions that decide whether patterns match. Third, the set of actions in the bodies of rules can vary. Note that if a body must always remove the elements that matched its pattern from the database and can otherwise only add structures, the production system can be run using a backward-chaining control structure.

The final source of variety in production systems is the conflict-resolution mechanism that is used to deal with situations in which more than one rule matches the database. MYCIN (33), e.g., simply executes all of the matching rules. A strict Markovian system, on the other hand, would simply choose the first rule in some ordering of the rules. Systems such as OPS-5 (qv) (34) have an extensive set of heuristics that use such factors as the complexity of the patterns and the relative ages of competing rules to choose in the case of conflicts. The systems described in Ref. 11 exhibit a variety of additional conflict-resolution schemes.

**FRL.** FRL (qv) (18) is a fairly straightforward adaptation of Minsky's frame proposal. The basic unit of representation is the frame. Frames are organized into a hierarchy by the relationship called AKO ("a kind of"). This relationship combines both generalization (a student is a kind of person) and classification (John is a kind of person). A frame is a structured object consisting of a set of slots each of which has one or more of the following items:

comments describing the source of the value; e.g., the comment associated with John's student number might be "typed in by George in response to information supplied by the registrar's office";

a default value; this is returned when no value has been specified for the slot. This is most interesting when the slot is inherited. For example, if the slot student-number is

given a default value of 741353839 in the class STUDENT and John is defined as a kind of STUDENT, John will inherit the slot student-number and if no value is provided, any attempt to access the student number of John will result in the default value (see Reasoning, default);

a constraint; this is an expression that must be true of a candidate value for the slot. The age slot, e.g., might have the constraint "(greater-than 0)";

a procedure to access the value of the slot. Suppose the frame student has a slot called list-of-courses. Then in defining the slot number-of-courses, one could use an access procedure that computes the number of courses by counting the items that are in the list of courses;

a procedure to store the value of the slot. For example, storing a value in the list-of-courses might require a procedure that adds John to the students slot of each course in the list; and

a procedure to remove the value of the slot; when the list of courses is removed, John must be removed from the students slot of each course.

FRL provides no explicit inference mechanisms. Inference must be encoded in the procedures associated with slots and the programs using the FRL knowledge base.

**KRL.** KRL (qv) (17) is another frame-based knowledge representation language. Unlike FRL, KRL supplies an inference mechanism in the form of pattern matching. The basic unit of representation is again called a frame and is a structured object consisting of slots. KRL provides a notation that includes primitives for many of the common ways of forming objects. For example, one might define Socrates as (a persons with name = Socrates).

Implicit knowledge is retrieved by offering a pattern to the pattern matcher. A complex pattern is broken up into subpatterns that are placed on an agenda of matches to perform (see Agenda-based systems). The order of execution of these matches may be controlled by the user of the formalism. A slot may be associated with a LISP procedure that specifies how to match a value. This procedure may itself add match requests to the agenda. KRL allows frames to be placed into partitions called belief contexts, which serve to focus the attention of the matching process.

**KL-ONE.** Another important representation language is KL-ONE (qv) (35). A KL-ONE knowledge base consists of concepts that are descriptions. The slots of a concept are called roles. As usual, slots may have defaults, constraints, attached procedures, etc. An important contribution, however, is the introduction of an explicit representation for constraints involving more than one role. This is done using structural descriptions. For example, to represent the fact that the enrollment date in school of a person precedes the graduation date, one would use a structural description that points to the roles enrollment-date and graduation-date and to the concept precedes, which is a description of the precedence relationship.

KL-ONE makes strong use of the specialization relationship. Although there is no classification, there is a distinction between concepts describing sets of objects and concepts describing individuals. The concepts that describe individual objects are called individual concepts, and their relationship to generic concepts is called individuation.

A fact about the world is asserted through the creation of an object called a nexus. For example, a nexus linked to a concept describing some person means that there exists a person which satisfies all of the properties associated with the concept. Nexi can be asserted in various contexts corresponding to differing models of the world.

**Other Systems.** OWL (36) is a frame-based scheme based on the syntactic and semantic structure of English: the unit of representation is the node, which is an expression representing a sentence of English.

Other systems include PSN (19), a frame-based system that uses attached procedures to manage the instances of classes; OMEGA (37), which provides a calculus of descriptions; and KRYPTON (20), which combines a frame-based component for forming descriptions and a logic-based component for asserting facts.

### Current Research

**Incompleteness.** Except for the simplest of problems, it is unlikely that a knowledge base will contain everything that there is to be known about a given domain. For example, suppose a knowledge base containing information about students and courses is asked "Is George a philosopher?" A common response is to answer no if the answer cannot be inferred. In effect, failure to find an answer is assumed to be equivalent to finding the negation of the query. This assumption is called the closed-world assumption (CWA) (38). Ideally, however, the knowledge base should respond that the answer cannot be inferred from its facts. In addition, it should be possible to ask the knowledge base about the extent of its knowledge.

The preceding query is an example of incompleteness due to lack of information. The knowledge base may also be incomplete if the inference engine is too weak to derive the desired information (21). On the other hand, Gödel's incompleteness result guarantees that when the system is sufficiently powerful, there will always be true facts that cannot be derived from the knowledge base. A theory in first-order logic that includes arithmetic is an example of an incomplete system.

Levesque (39) and Moore (40) suggest approaches toward dealing with incompleteness. Levesque, introduces a new operator  $K$  that when placed in front of a fact is read as "it is believed that. . . ." It is then possible to ask the knowledge base

$K$  philosopher(George)

that is, is it known that George is a philosopher. Levesque's work analyzes the  $K$  operator and addresses the computational consequences of a language having this operator.

**Nonmonotonicity and Defaults.** When a knowledge base is incomplete, it is often necessary to make assumptions in order to arrive at an answer. For example, one might have an instance of the class bird but not know what kind of bird it is. It is often reasonable to assume that the bird can fly and to make inferences using this assumption. Such assumptions are often stored in knowledge bases as default values for certain properties (e.g., Ref. 28). Defaults raise a difficult problem. Suppose it is later discovered that the bird is in fact a penguin. Any inferences arising from the assumption that the bird could fly are now invalid. This differs from knowledge bases using conventional logics where new knowledge never causes the re-

traction of facts deduced from previously available knowledge. Schemes that have the latter property are called monotonic (see Reasoning, default; Reasoning, nonmono-tonic).

There have been several attempts to deal rigorously with nonmonotonic schemes. Among these is a default logic (38) in which inference rules of the form "if it is consistent to assume  $X$  then assume  $X$ " are allowed. Another is circumscription (qv) (41), which provides a schema for generating statements that say things such as "all that is true about  $X$  is all that is known about  $X$ ."

A related problem is that of inconsistent knowledge bases. For example, if one makes an inference from a default and stores the resulting fact in the knowledge base, a contradiction will result if the default is later found to be wrong. Inconsistency may also result if new knowledge simply contradicts information previously entered into the knowledge base. In conventional logics, inconsistency is a very serious problem because everything is derivable from (and hence implicitly in) the knowledge base. One approach to inconsistency is belief revision (42) in which the knowledge-based system selectively removes statements that result in contradictions.

**Uncertainty and Imprecision.** A common problem in expert systems is that the knowledge provided by experts is uncertain. That is, statements are qualified by caveats such as "I am 90% sure." One must then determine the certainty of derived facts. This has proven to be a difficult problem to formalize. Many expert systems [e.g., PROSPECTOR (qv)] have successfully used simplifications of Bayes's rule from probability (see Bayesian decision methods), which gives a formula for the certainty of a derived fact in terms of the certainties of the facts used in the derivation and probabilities of interactions of these facts. The probabilities of interactions are usually very difficult to acquire; one of the first simplifications is therefore to assume that there are no interactions.

An orthogonal problem is imprecision: This is illustrated by statements such as "most Swedes are blond." Fuzzy sets (43) have been suggested as a way of dealing with imprecision.

### Conclusion

Although there is little agreement as to what knowledge representation actually is, many schemes have been proposed as general frameworks for representing and storing knowledge. Some have been successfully used as a basis for working systems. There are, however, many features of knowledge such as defaults that are not well understood. Until a better understanding of these features is reached, knowledge representation will remain an active area of study (e.g., see Ref. 44).

### BIBLIOGRAPHY

1. R. J. Brachman and B. C. Smith, (guest eds.), "Special issue on knowledge representation," *SIGART Not.* **70**(1980).
2. G. McCalla and N. Cercone (guest eds.), *IEEE Comput.* (Special Issue on Knowledge Representation) **16**(10) (October 1983).
3. A. Barr and J. Davidson, Representation of Knowledge, in A. Barr and E. A. Feigenbaum (eds.), *The Handbook of Artificial Intelligence*, Vol. 1, W. Kaufmann, Los Altos, CA, pp. 140-221, 1981.
4. J. Mylopoulos and H. J. Levesque, An Overview of Knowledge Representation, in M. L. Brodie, J. Mylopoulos, and J. W. Schmidt

- (eds.), *On Conceptual Modelling*, Springer-Verlag, New York, pp. 3–18, 1984.
5. J. McCarthy, Programs with Common Sense, in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, pp. 403–418, 1968.
  6. P. J. Hayes, In Defense of Logic, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, pp. 559–65, 1977.
  7. M. Minsky (ed.), in *Semantic Information Processing*, MIT Press, Cambridge, MA, 1968.
  8. M. R. Quillian, Semantic Memory, in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, pp. 227–270, 1968.
  9. A. Newell and H. A. Simon, *Human Problem Solving*, Prentice Hall, Englewood Cliffs, NJ, 1972.
  10. F. Hayes-Roth, D. A. Waterman, and D. B. Lenat (eds.), *Building Expert Systems*, Addison-Wesley, Reading, MA, 1983.
  11. D. A. Waterman and F. Hayes-Roth (eds.), *Pattern-Directed Inference Systems*, Academic, London, 1978.
  12. C. Hewitt, PLANNER: A Language for Proving Theorems in Robots, *Proceedings of the Second International Joint Conference on Artificial Intelligence*, London, pp. 167–181, 1971.
  13. G. J. Sussman and D. V. McDermott, Why Conniving is Better than Planning, Technical Report MIT-AI-225A, Massachusetts Institute of Technology, April 1972.
  14. W. A. Woods, What's in a Link: Foundations for Semantic Networks, in D. G. Bobrow and A. Collins (eds.), *Representation and Understanding*, Academic Press, New York, pp. 35–82, 1975.
  15. N. V. Findler (ed.), *Associative Networks: Representation and Use of Knowledge by Computers*, Academic Press, New York, 1979.
  16. M. Minsky, A Framework for Representing Knowledge, in P. H. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, pp. 211–277, 1975.
  17. D. G. Bobrow and T. Winograd, "An overview of KRL, a knowledge representation language," *Cog. Sci.* 1, 3–46 (1977).
  18. I. P. Goldstein and R. B. Roberts, Nudge: a Knowledge-Based Scheduling Program, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, pp. 257–263, 1977.
  19. H. J. Levesque and J. Mylopoulos, A Procedural Semantics for Semantic Networks, in N. V. Findler (ed.), *Associative Networks: Representation and Use of Knowledge by Computers*, Academic Press, New York, pp. 93–120, 1979.
  20. R. J. Brachman, R. E. Fikes, and H. J. Levesque, "Krypton: A functional approach to knowledge representation," *IEEE Comp.* 16(10), 67–74 (October 1983).
  21. H. J. Levesque, A Fundamental Tradeoff in Knowledge Representation and Reasoning, *Canadian Society for Computational Studies of Intelligence/Société Canadienne pour l'Étude de l'Intelligence par Ordinateur, Proceedings of the Fifth Biennial Conference*, London, Ontario, pp. 141–152, 1984.
  22. A. Tarski, *Logic, Semantics, Metamathematics*, Oxford University Press, New York, 1956.
  23. D. J. Israel and R. J. Brachman, Distinctions and Confusions: A Catalogue Raisonné, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B. C. pp. 452–459, 1981.
  24. L. K. Schubert, "Extending the expressive power of semantic networks," *Artif. Intell.* 7, 163–198 (1976).
  25. G. G. Hendrix, Encoding Knowledge in Partitioned Networks, in N. V. Findler (ed.), *Associative Networks: Representation and Use of Knowledge by Computers*, Academic Press, New York, pp. 51–92, 1979.
  26. J. A. Robinson, *Logic: Form and Function*, Edinburgh University Press, Edinburgh, UK, 1979.
  27. H. Ait-Kaci, A New Model of Computation Based on a Calculus of Type Subsumption, MS-CIS-83-40, Department of Computer and Information Science, The Moore School of Electrical Engineering, University of Pennsylvania, 1983.
  28. S. E. Fahlman, *NETL: A System For Representing and Using Real World Knowledge*, MIT Press, Cambridge, MA, 1979.
  29. R. Davis, "Meta-rules: Reasoning about control," *Artif. Intell.* 15(3), 179–222 (December 1980).
  30. B. M. Kramer, Control of Reasoning in Knowledge-Based Systems, Ph.D. Thesis, University of Toronto, Toronto, Ontario, 1986.
  31. R. Kowalski, *Logic for Problem Solving*, Elsevier North Holland, New York, 1979.
  32. L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy, "The HEARSAY-II speech-understanding system: Integrating knowledge to resolve uncertainty," *ACM Comput. Surv.* 12(2), 213–253 (June 1980).
  33. E. H. Shortliffe, *Computer-Based Medical Consultations: MYCIN*, American Elsevier, New York, 1976.
  34. J. McDermott and C. Forgy, Production System Conflict Resolution Strategies, in D. A. Waterman and F. Hayes-Roth (eds.), *Pattern-Directed Inference Systems*, Academic Press, New York, pp. 177–199, 1978.
  35. R. J. Brachman, On the Epistemological Status of Semantic Networks, in N. V. Findler (ed.), *Associative Networks: Representation and Use of Knowledge by Computers*, Academic Press, New York, 1979.
  36. W. A. Martin, Descriptions and the Specializations of a Concept, in Winston and Brown (eds.), *Artificial Intelligence: An MIT Perspective*, MIT, Cambridge, MA, pp. 379–419, 1979.
  37. C. Hewitt, G. Attardi, and M. Simi, Knowledge Embedding in the Description System Omega, *Proceedings of the First National Conference on Artificial Intelligence*, Stanford, CA, pp. 157–164, 1980.
  38. R. Reiter, On Closed World Data Bases, in H. Gallaire and J. Minker (eds.), *Logic and Databases*, Plenum, New York, pp. 55–76, 1978.
  39. H. J. Levesque, A Logic of Implicit and Explicit Belief, in *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, pp. 198–202, 1984.
  40. R. C. Moore, Reasoning About Knowledge and Action, Technical Note 284, AI Centre, SRI International, Palo Alto, CA, 1980.
  41. J. McCarthy, "Circumscription—A form of non-monotonic reasoning," *Artif. Intell.* 13, 27–39 (1980).
  42. J. Doyle and P. London, "A selected descriptor-indexed bibliography to the literature of belief revision," *SIGART Newslet.* 71, 7–23 (April 1980).
  43. L. A. Zadeh, Commonsense Knowledge Representation Based on Fuzzy Logic, ERL Memo M83/26, University of California, Berkeley, 1983.
  44. K. Parsaye and D. Flenniken, "A Uniform Approach to Knowledge Representation and Inference," **IntelligenceWare Technical Report 9/6/86**, IntelligenceWare, Inc., Los Angeles, CA, 1986.

B. M. KRAMER AND J. MYLOPOULOS  
University of Toronto

**REPRESENTATION, PROCEDURAL.** See Planner; Rule-based systems.

## REPRESENTATION, WIRE-FRAME

Computers are able to create graphic screen images easily. These images may represent two- and three-dimensional ob-



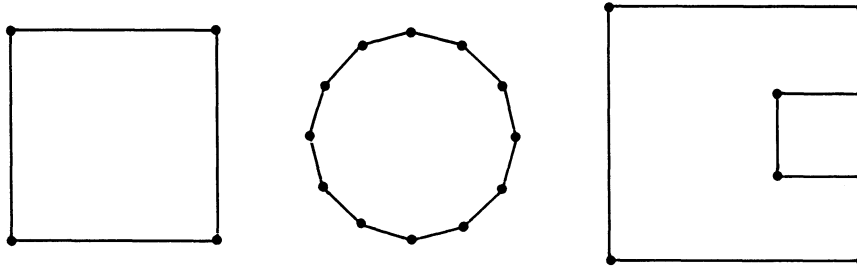


Figure 1. Two-dimensional wire-frame representation.

jects. In creating two-dimensional images the computer screen is divided horizontally into the  $X$  axis and vertically into the  $Y$  axis. The vertices of the two-dimensional object are plotted on the screen by using their  $XY$  coordinates and then these points are connected with straight-line segments (Fig. 1). In this way any shape may be drawn on the computer screen.

Three-dimensional objects may be represented in a wire-frame manner on the computer screen. This system also uses points in space to represent the vertices of the object. These points are then connected with lines that form the edges of the object (Fig. 2).

In order to describe these points, one of two variations of Cartesian coordinates are used. The first variation has positive  $Z$  values coming out of the screen from the origin. This is called the right-hand system (RHS). The second has negative  $Z$  values projecting out of the screen from the origin and is referred to as the left-hand system (LHS) (Fig. 3). In both systems the  $X$  axis is horizontal and positive as it moves to the right of the origin and the  $Y$  axis is vertical and positive as it moves up from the origin.

No matter which system is used, a point must be described by an  $X$ , a  $Y$ , and a  $Z$  value. After these points have been described, the list of connecting lines is developed. The combination of the list of points and list of lines forms the database that represents the object (Table 1). This table is the database for the object in Figure 2.

After a database is developed, it is used to map the three-dimensional primitive on the computer screen. A number of transformations may be carried out with this database. These include translations (moving) of the wire-frame left/right, up/down, and in/out along the  $X$ ,  $Y$ , or  $Z$  axis. Scaling is another

transformation that may be carried out. Scaling the object along all three axes at the same time will give a proportionally larger or smaller version of the original object (Fig. 4).

If the wire frame is scaled along only one axis, a distorted or stretched version of the object with a different aspect ratio will result (Fig. 5). In this figure is seen the original object and the same object scaled only along the  $Y$  axis.

This database may also be manipulated to rotate the object about the  $X$ ,  $Y$ , or  $Z$  axis or all three axes at once (Fig. 6). This gives the illusion of walking around the object or that the object is turning in space. Figure 6a shows a frontal view of the object. Figure 6b is the same object rotated  $45^\circ$  about the  $Y$

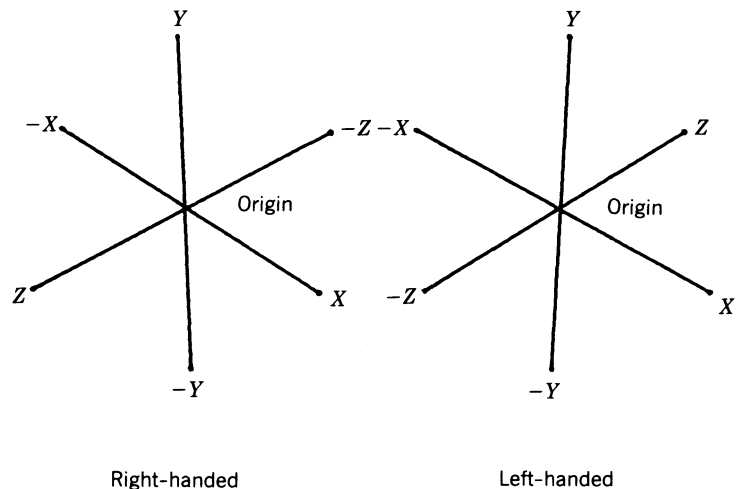


Figure 3. Coordinate system.

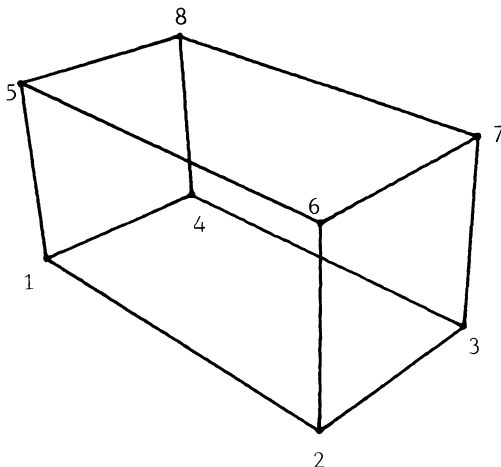


Figure 2. Three-dimensional wire-frame representation.

Table 1. Object Database

Points	$X$	$Y$	$Z$	Lines	From Point	To Point
1	0	0	0	1	1	2
2	4	0	0	2	2	3
3	4	0	2	3	3	4
4	0	0	2	4	4	1
5	0	2	0	5	5	6
6	4	2	0	6	6	7
7	4	2	2	7	7	8
8	0	2	2	8	8	5
				9	1	5
				10	2	6
				11	3	7
				12	4	8

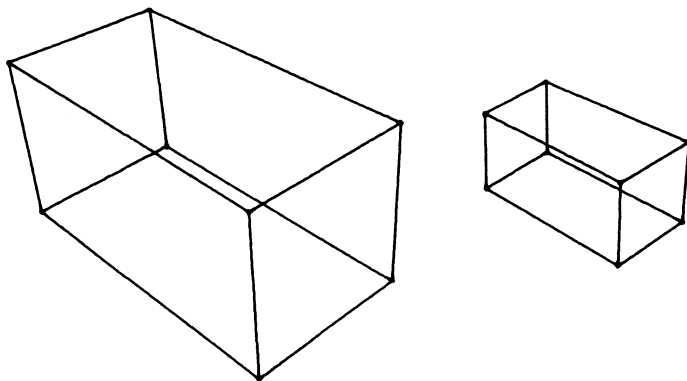


Figure 4. Scaled object.

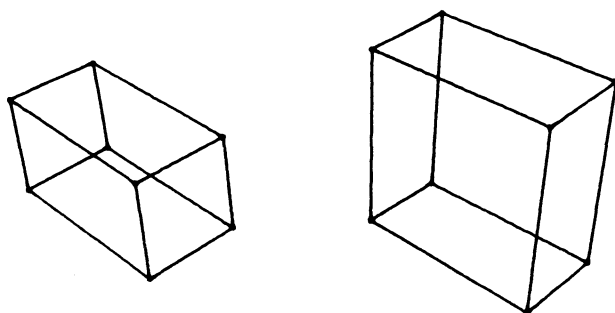


Figure 5. Distorted object.

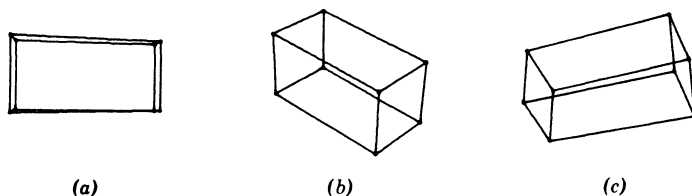


Figure 6. Rotated objects.

axis and  $35^\circ$  about the  $X$  axis. Figure 6c shows the object in  $b$  rotated  $45^\circ$  about the  $Z$  axis.

Wire-frame representations may be as simple as the previous examples or very complex as in (Fig. 7).

The value of the database is not limited to creating and manipulating the screen images. Databases of this type are at the heart of many CAD/CAM systems (see Computer-aided design; Computer-integrated manufacturing) and allow other operations including high-quality dimensioned drawings, numerical control machining, and computer-aided quality control. More important, these wire-frame images could graphically represent complex relationships, processes, or environments. By combining these computer images with the knowledge base of the AI expert systems, a more powerful tool will develop. This will allow the user to converse with the expert system using symbols as well as words. This could more easily guide the user through analytical and diagnostic processes.

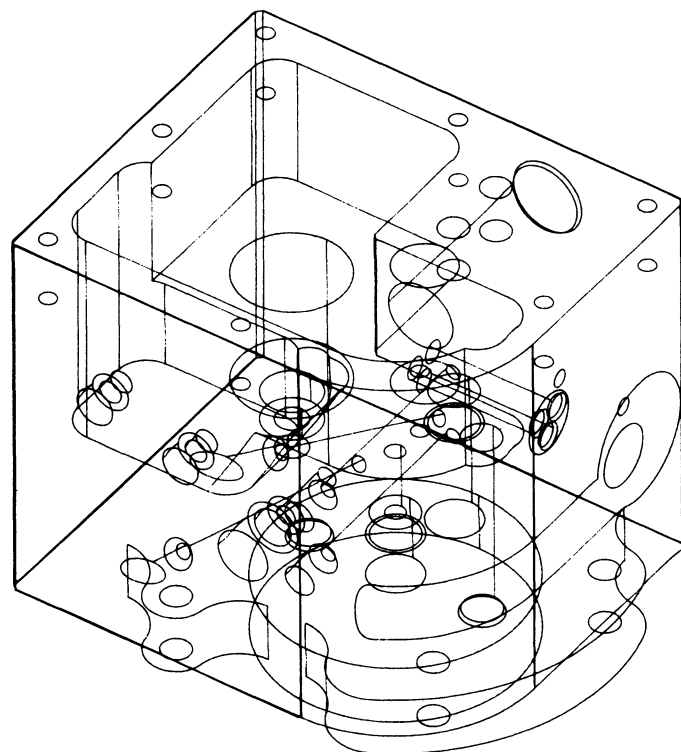


Figure 7. Housing (courtesy of Gary Rafe).

#### General References

- S. Harrington, *Computer Graphics A Programming Approach*, McGraw-Hill, New York, 1983.
- D. Ryan, *Computer Programming for Graphical Displays*, Brooks/Cole Engineering Division, Wadsworth, Belmont, CA, 1983.
- J. Demel and M. Miller, *Introduction to Computer Graphics*, Brooks/Cole Engineering Division, Wadsworth, Belmont, CA, 1984.
- B. Artwick, *Applied Concepts in Microcomputer Graphics*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

R. KEOUGH  
Rochester Institute of Technology

RESEARCH METHODOLOGY. See Philosophical questions.

#### RESOLUTION, BINARY: ITS NATURE, HISTORY, AND IMPACT ON THE USE OF COMPUTERS

Resolution, or, more accurately, binary resolution (1), is a formal inference rule whose use permits computer programs to "reason." Its formulation has had an important effect on the use of the computer. Binary resolution considers two statements satisfying appropriate conditions and deduces a third statement that follows logically from them. As the examples of the next two sections show, a computer program relying on resolution and related inference rules can find new facts and new relationships by "reasoning" (qv) from given facts and relationships. Such "automated reasoning programs" (2-4) are currently (in 1985) being used for different types of research and diverse applications. They have provided valuable assistance in answering previously open questions from math-

ematics (5–7) and logic (8,9), in designing superior logic circuits (10), in validating existing circuit designs (11), and in verifying certain properties claimed for existing computer programs. Were it not for the formulation of resolution and the research it spawned, such programs—and even the field of automated reasoning itself—might not exist today.

To compare a resolution-based approach with the approach that preceded it, a brief overview is needed. Then, so that resolution itself can be understood and the significance of the research stemming from its formulation appreciated, this entry turns to the following topics: an informal treatment of how resolution works; the formal definitions and properties of resolution and related inference rules; strategies for directing and strategies for restricting the application of inference rules; various reasoning programs; current applications; and a brief summary.

By 1960 various logicians and mathematicians had expressed interest in the objective of using the computer to prove theorems from diverse areas of mathematics (see Theorem proving). Computer programs (12–15) had been implemented that could prove extremely simple theorems. The most popular approach at the time was to use a particular language of logic (see below) and rely on Herbrand instantiation (16,17) (see below). Rather than deducing new facts and relationships from given statements, the Herbrand instantiation approach draws conclusions by merely substituting for the variables in the given statements to yield trivial consequences of them. In contrast, applications of resolution can yield new information, and hence the reasoning it affords is more substantial. With both Herbrand instantiation and resolution, the object is to consider some given statement corresponding to a purported theorem of the form “if  $P$ , then  $Q$ ” and attempt to prove that the statement is in fact a theorem. With either approach, rather than the given statement, the statement actually considered corresponds to assuming  $P$  true and  $Q$  false. This choice is made because the approach seeks a “proof by contradiction.”

For a brief illustration of the instantiation approach, a simple theorem from abstract algebra will suffice—an example, incidentally, that is beyond the power of those early theorem-proving programs that depend on instantiation. A favorite example from 1965 until this writing (1985) is the theorem “in a group, if the square of every element is the identity, then the group is commutative.” Let the theorem be represented symbolically as “if  $P$ , then  $Q$ ,” where  $P$  consists of the set of axioms for a group together with the special hypothesis that the square of every element is the identity, and where  $Q$  is the conclusion that states that the group is commutative. The instantiation procedure begins by assuming the conclusion  $Q$  false and therefore replaces  $Q$  with the denial or negation of the conclusion. In particular, it is assumed that (at least) two elements  $a$  and  $b$  exist such that the product  $ab$  is not equal to the product  $ba$ . The procedure next calls for the statements comprising  $P$  and those resulting from assuming  $Q$  false all to be represented in the “clause language.” For example, the clause

$$\text{EQUAL}(\text{prod}(\text{prod}(x, y), z), \text{prod}(x, \text{prod}(y, z)))$$

is an acceptable way to express the associativity of product,  $(xy)z = x(yz)$  for all  $x, y$ , and  $z$ . Let  $S$  denote the set of clauses that is thus produced. Then, by substituting (Herbrand) constants into the clauses comprising  $S$ , the procedure generates ever-expanding sets of variable-free clauses in search of a set

that can be proved truth-functionally unsatisfiable. A set is “truth-functionally unsatisfiable” if all consistent assignments of **true** or **false** to the elements that make up the variable-free clauses in the set yield **false**. The basic result from logic (16,17) on which the procedure rests says that if the starting set  $S$  of clauses does in fact correspond to assuming an actual theorem false, one of the variable-free sets of clauses considered by the instantiation procedure is truth-functionally unsatisfiable, and conversely (as shown below, resolution also relies on the same basic result).

As it turns out, this procedure of considering ever-expanding sets obtained by Herbrand instantiation and then testing for truth-functional unsatisfiability has two undesirable properties. First, the sets of variable-free clauses become very large very early in the consideration, which in turn causes the test for unsatisfiability to become very cumbersome. Second, the generality that often contributes to efficiency in searching for a proof is lacking. In particular, the procedure must cope with many instances of a fact rather than coping with the fact itself. For example, if the purported theorem under investigation were about arithmetic, the procedure under discussion might be forced to consider

$$0 + a = a$$

$$0 + (a + a) = (a + a)$$

$$0 + -a = -a$$

$$\vdots$$

rather than coping with  $0 + x = x$  (for all  $x$ ), which captures the preceding instances as trivial corollaries.

Computer programs employing this Herbrand instantiation procedure are not able to prove very interesting theorems. In addition to offering only the most elementary type of reasoning, they suffer from employing a level-saturation search for the desired unsatisfiable set. (The search is called level saturation because all Herbrand constants of level 1 are first considered for substitution, then those of level 2, etc.)

By formulating the inference rule of binary resolution (1), Robinson provided an important advance over the Herbrand instantiation approach. What he did, directly and indirectly, was to provide a means for computers to reason far more effectively than was previously possible. Resolution focuses on general statements rather than on trivial instances of them. In addition, its use avoids truth-functional analysis. Finally, and perhaps most important, Robinson's work led to the formulation of more effective inference rules (18–27) and to the discovery of powerful strategies (28–30) to control the application of inference rules. Such research was prompted by the observation that an approach relying solely on binary resolution is simply not effective for solving most problems. Its application yields too many conclusions, each representing too small a step of reasoning. The need for inference rules that take larger reasoning steps and for strategies to control the reasoning can easily be established with experimentation (26). The significance of the research triggered by the formulation of resolution is amply demonstrated by the successes achieved with a number of automated reasoning programs (2,3,4,31–34) which are based on the results of that research (logic programming, and especially Prolog, can be traced to the formulation of resolution; see Logic programming). The fact that the successes range from answering previously open questions (5–9) to de-

signing superior logic circuits (10) establishes that resolution-based reasoning programs are useful in a number of unrelated areas.

From the historical perspective, the research that was well under way by 1960 caused an entire field to come into existence. Robinson's contribution of resolution represented an important step forward for that field. (Although the detailed paper on resolution was not published until 1965, Robinson formulated the rule in 1963.) The objective of "proving theorems" with a computer program gave the name to the field at the time. It was first called "mechanical theorem proving," then "automatic theorem proving," and finally by 1968 "automated theorem proving" (see Theorem proving). In 1980 noting that, in addition to finding proofs, the uses of such computer programs include finding models and counterexamples (7), testing hypotheses, and designing circuits, Wos introduced the name "automated reasoning" (35) for the expanded field. Because of the excitement and interest engendered by this new field of automated reasoning, and because resolution plays such a vital role, resolution-based reasoning programs will continue to be the focus of attention for the foreseeable future.

### Informal Treatment of Binary Resolution

To gain a feeling for the actions of binary resolution, recall how both *modus ponens* and syllogism work. *Modus ponens* applied to the (variable-free) statements

Nan is female

and

Nan is **not** female **or** Nan is **not** male

yields

Nan is **not** male

as the conclusion. Syllogism applied to the (variable-free) statements

Nan is French **or** Nan is Irish

and

Nan is **not** Irish **or** Nan is Spanish

yields

Nan is French **or** Nan is Spanish

as the conclusion. If binary resolution is applied to either of the two examples, where the examples are expressed in the clause language, the conclusions just cited will also be drawn. For example, if the clauses

FEMALE(Nan)

$\neg$ FEMALE(Nan) |  $\neg$ MALE(Nan)

are considered simultaneously by resolution, then the clause

$\neg$ MALE(Nan)

is obtained as the logical conclusion, where  $\neg$  is read as **not** and | is read as **or**. The conclusion is obtained by "canceling" FEMALE(Nan) in the first clause with  $\neg$ FEMALE(Nan) in the second.

In addition to "canceling" part of one clause against part of

another, binary resolution relies on "substituting" expressions for variables within a clause. For example, where the symbol  $x$  is a variable, the clause

$\neg$ FEMALE( $x$ ) |  $\neg$ MALE( $x$ )

says that " $x$  is **not** female **or**  $x$  is **not** male" for (implicitly) all  $x$ . If binary resolution considers this clause simultaneously with the earlier clause

FEMALE(Nan)

again the conclusion is

stating that Nan is **not** male. In effect, the conclusion is obtained by first substituting Nan for the variable  $x$  to obtain the temporary clause

$\neg$ FEMALE(Nan) |  $\neg$ MALE(Nan)

and then canceling as before. Although the actions of binary resolution for this last example have been explained as a two-step process, in reality the inference rule has the useful property of drawing conclusions in one step without generating such temporary clauses. (The name "binary resolution" suggests the nature of the inference rule. First, the rule always simultaneously considers two clauses from which to attempt to draw a conclusion, hence "binary." Second, the rule attempts to resolve the question of what is the maximum common ground covered by obvious consequences of the two clauses, hence "resolution." In the first of the two examples of applying binary resolution, no effort is required to find the common ground; no substitution for variables is necessary to permit an obvious cancellation. In the second example, however, to find the common ground requires substituting Nan for the variable  $x$  in the second clause.)

A more important property of resolution is the generality of the conclusions yielded by its application. For example, from the clauses

$\neg$ HASAJOB( $x$ , nurse) | MALE( $x$ )

(for all  $x$ ,  $x$  is **not** a nurse **or**  $x$  is male)

$\neg$ FEMALE( $y$ ) |  $\neg$ MALE( $y$ )

(for all  $y$ ,  $y$  is **not** female **or**  $y$  is **not** male)

binary resolution yields the clause

$\neg$ HASAJOB( $x$ , nurse) |  $\neg$ FEMALE( $x$ )

(for all  $x$ ,  $x$  is not a nurse **or**  $x$  is not female)

as the conclusion. It does not yield

$\neg$ HASAJOB(Kim, nurse) |  $\neg$ FEMALE(Kim)

(Kim is not a nurse **or** Kim is not female)

which also follows logically from the two given clauses. This last conclusion simply lacks the generality that is so vital to the effectiveness of the various automated reasoning programs now in use.

In addition to the "generality" property, binary resolution has a vital logical property. The inference rule is "sound"—the conclusions drawn with it follow inevitably from the two statements to which it is applied. Thus, the arguments produced by relying solely on resolution are flawless; any false piece of information must come from a faulty hypothesis. To comple-

ment the property of soundness, a desirable property possessed by some inference rules (or some sets of inference rules) is that of "refutation completeness." If an inference rule is refutation complete, there exists a procedure employing that inference rule that, when presented with a set of clauses corresponding to the denial (or negation) of a theorem, guarantees to find a proof of the inconsistency of that set eventually. Binary resolution, if augmented by the inference rule of "factoring" (see below), has that property.

Unfortunately, possession of the properties of soundness and refutation completeness does not imply that the inference rule (or set of inference rules) is effective. In fact, binary resolution (even if augmented with factoring) is not effective in most problem domains. The inference rule usually yields too many conclusions, each representing too small a step of reasoning. Nevertheless, programs employing binary resolution and factoring as their sole inference rules are far superior to those relying on Herbrand instantiation. Resolution-based programs do actually "reason" rather than merely drawing trivial conclusions by instantiating (substituting for variables) the given information. In particular, they deduce new facts and new relationships from existing ones. Since the information deduced with resolution exhibits the generality property discussed above, the formulation of resolution represents an important step forward in the quest for computer programs that could and would find proofs of purported theorems. That quest in fact culminated in the implementation of very useful programs (see below). Those programs rely on research that can be directly traced to the discovery of binary resolution. The results of such research include the formulation of far more effective inference rules and the discovery of powerful strategies for controlling the inference rules.

### Formal Treatment of Binary Resolution and Related Inference Rules

Binary resolution and related inference rules possess important logical properties. These properties contribute to the impetus for using such formal rules. Not only are the arguments produced with such rules logically sound, but when certain conditions are met, their use guarantees that a "proof by contradiction" can be found. To give the precise conditions and discuss "proof by contradiction," certain theorems from logic and various formal definitions are required. Since inference rules are designed to draw conclusions—to operate on statements to produce other statements—a discussion of a language sufficient for representing many problems is also needed.

**Representation.** The most commonly used language for representing information when using binary resolution and related inference rules is the "clause language," a language directly descended from the pure first-order predicate calculus (see Logic, predicate). The following definitions briefly formalize the clause language.

**Definition.** A "term" is a constant, a variable, or an  $n$ -ary function symbol followed by  $n$  arguments all of which are terms.

**Definition.** An "atom" is an  $n$ -ary predicate symbol followed by  $n$  arguments all of which are terms.

**Definition.** A "literal" is an atom or the negation of an atom.

**Definition.** A "clause" is a finite disjunction of zero or more distinct literals. All variables in a clause are (implicitly) universally quantified, and their scope is just the clause in which they occur.

**Definition.** A "clause set" is the (implicit) conjunction of a set of clauses.

For example, a clause set might consist of the following three clauses.

$$\begin{aligned} &\neg \text{FEMALE}(x) \mid \neg \text{MALE}(x) \\ &\text{FEMALE}(x) \mid \text{MALE}(x) \\ &\neg \text{HASAJOB}(x, \text{nurse}) \mid \text{MALE}(x) \end{aligned}$$

An implicit logical **and** exists between the first and second clauses and also between the second and third. Although the variable  $x$  occurs in all three clauses, replacing  $x$  by the variable  $y$  in the second and replacing  $x$  by the variable  $z$  in the third produces a logically equivalent set of clauses. (Clauses are treated as having no variables in common.) The literal  $\neg \text{MALE}(x)$  is the negation of the atom  $\text{MALE}(x)$ , which itself is a literal.

Statements can be represented in the clause language by first representing them (where possible) with a formula in the first-order predicate calculus and then transforming them with a well-known procedure (36). The procedure first transforms a formula to prenex normal form and then to conjunctive normal form and finally removes each existentially quantified variable by instead employing an appropriate function or constant. The functions and constants that are employed are called "Skolem functions" (14). The remaining quantifiers are then dropped to yield the set of clauses. Although the procedure does not produce a logically equivalent formula, the needed logical properties for proof finding are preserved (see below). For example, the clause

$$\text{EQUAL}(\text{sum}(y, \text{minus}(y)), 0)$$

is an acceptable translation of the statement "there exists  $z$  such that for all  $y$  there exists  $x$  such that the sum of  $y$  and  $x$  equals  $z$ ." This clause employs the Skolem function *minus* and the Skolem constant (function of no variables) *0*, and, although not logically equivalent to the formula from which it is obtained, it conveys the required meaning. The clause language does not accept statements employing such logical operators as **if-then** and **equivalent** but instead relies on transformations that replace such statements with others that are logically equivalent to them and that use **not**, **or**, and (implicitly) **and**.

**Proof Finding.** The most common use of a resolution-based computer program is to find a proof of some purported theorem. The typical proof found by such a program is a "proof by contradiction." To seek such a proof, a formula in the first-order predicate calculus is written (if possible) that corresponds to assuming the purported theorem false. As discussed above, the formula is then transformed into a corresponding set of clauses. The definitions and theorems described here establish that if the first-order formula does in fact represent the denial (or negation) of a theorem, the resulting set of clauses possesses the needed logical properties to permit a proof by contradiction to be found. Before turning to the pre-

cise formalism, a trivial example from above serves as an illustration.

The specific fact that Nan is female and the general fact that everyone is **not** female **or not** male together obviously imply that Nan is **not** male. If the goal were to have a reasoning program find a proof of this obvious conclusion, the approach calls for assuming the conclusion false. From the corresponding clauses

1. FEMALE(Nan)
2.  $\neg \text{FEMALE}(x) \mid \neg \text{MALE}(x)$
3. MALE(Nan)

a one-step proof can easily be obtained. By applying binary resolution to clauses 2 and 3,

4.  $\neg \text{FEMALE}(\text{Nan})$

is deduced, which obviously contradicts clause 1. Clause 1 is the special hypothesis, clause 2 the only axiom, and clause 3 the denial (or negation) of the conclusion. With this simple example of a proof by contradiction in view, the formalism that justifies the underlying approach can now be given with the following definitions:

**Definition.** An "interpretation" of a variable-free set of clauses is an assignment of either **true** or **false** to each of the literals in the clause set. The assignment must be consistent; if literal  $L$  is assigned the value **true**, then the negation of  $L$  must be assigned the value **false**.

**Definition.** A set of variable-free clauses is "truth-functionally satisfiable" if there exists an interpretation such that the conjunction of the clauses evaluates to **true**. A set of variable-free clauses is "truth-functionally unsatisfiable" if no interpretation exists that establishes the set to be truth-functionally satisfiable.

**Definition.** The "Herbrand universe" for a set  $S$  of clauses consists of all well-formed terms that can be composed from the function symbols and individual constants that are present in  $S$ . When no constants are present, the constant  $c$  is supplied.

**Definition.** An "Herbrand interpretation" of a set  $S$  of clauses is a consistent assignment of either **true** or **false** to each of the well-formed expressions that can be composed from the predicates of  $S$  and the terms from the Herbrand universe.

For the next definition, note that a set  $S$  of clauses, some of which may contain variables, can be evaluated to **true** or to **false** for a given Herbrand interpretation. Such an evaluation can be made by considering the full set of variable-free clauses obtainable by substituting terms from the Herbrand universe into the clauses in  $S$ . The full set is considered, for recall that the variables in a clause are (implicitly) assumed to mean "for all."

**Definition.** A set  $S$  of clauses is "satisfiable" if there exists an Herbrand interpretation of  $S$  for which  $S$  evaluates to **true**. A set  $S$  of clauses is "unsatisfiable" if no Herbrand interpretation exists that establishes the set to be satisfiable; in other words,  $S$  evaluates to **false** for every Herbrand interpretation.

The following definitions and theorem establish a much simpler criterion for determining the unsatisfiability of a set of clauses.

**Definition.** A "substitution" is a set of ordered pairs  $ti/vi$ , where the  $ti$  are terms and the  $vi$  are distinct variables.

**Definition.** The clause  $C'$  is an "instance" of the clause  $C$  if  $C'$  can be obtained from  $C$  by applying a substitution.

**Definition.** The clause  $C'$  is termed "a ground instance" of the clause  $C$  if  $C'$  is an instance of  $C$  such that  $C'$  is variable-free.

**Herbrand's Theorem.** A set  $S$  of clauses is unsatisfiable iff there exists a finite set of (Herbrand) ground instances of  $S$  that is truth-functionally unsatisfiable (35).

For example, the unsatisfiability of the set consisting of the clauses

$$P(a, y) \mid P(b, y) \\ \neg P(x, c)$$

can be established by considering the clauses

$$P(a, c) \mid P(b, c) \\ \neg P(a, c) \\ \neg P(b, c)$$

which comprise a truth-functionally unsatisfiable set of ground instances of the preceding set of two clauses. Note that there are two distinct instances of the second clause.

One final theorem is needed to justify the approach for finding proofs by relying on the use of the language of clauses.

**Theorem.** A formula of the first-order predicate calculus is unsatisfiable iff its representation in clause form is unsatisfiable (36).

**Binary Resolution and Related Inference Rules.** To define binary resolution, the following definitions are needed.

**Definition.** A "most general common instance" (MGCI), if one exists, of expressions  $E1$  and  $E2$  is an expression  $E$  such that  $E$  is an instance of both  $E1$  and  $E2$  and such that, if  $E'$  is an instance of both  $E1$  and  $E2$ , then  $E'$  is an instance of  $E$ .

**Definition.** The literals  $L1$  and  $L2$  are said to "unify" if there exists a substitution that, when applied to both  $L1$  and  $L2$ , produces  $L1'$  and  $L2'$  respectively such that  $L1' = L2'$ . The literals are then said to be "unifiable." A substitution that, when applied, yields an MGCI of two unifiable literals is called a "most general unifier" (MGU).

**Definition.** The inference rule "binary resolution" (1) yields the clause  $C$  from the clauses  $A$  and  $B$  (that are assumed to have no variables in common) when  $A$  contains the literal  $L1$ ,  $B$  contains the literal  $L2$ , one of  $L1$  and  $L2$  is a positive and the other a negative literal, and ignoring the sign,  $L1$  and  $L2$  are unifiable. The clause  $C$  is obtained by finding an MGU of  $L1$  and  $L2$  (ignoring the sign), applying the MGU to both  $A$  and  $B$  to yield  $A'$  and  $B'$ , respectively, and forming the disjunction of  $A' - L1'$  ( $A'$  minus a single occurrence of the literal  $L1'$ ) and  $B' - L2'$ . The clause  $C$  is termed a "resolvent" of  $A$  and  $B$ , and  $A$  and  $B$  are termed the "parents" of  $C$ .

For example, binary resolution applied to the two clauses

$$\neg P(x) \mid Q(x, y) \\ P(a) \mid P(f(b, z))$$

yields the two clauses

$$Q(a, y) \mid P(f(b, z)) \\ Q(f(b, z), y) \mid P(a)$$

as binary resolvents, depending on which literals are chosen in the first pair of clauses. On the other hand, binary resolution applied to the clauses



$$Q(y, y) \\ \neg Q(x, f(x))$$

yields no clauses. The two literals (ignoring the sign) cannot be unified; the two clauses have no common instance.

In order to be useful for proof finding, an inference rule should possess the logical properties of "soundness" and "refutation completeness."

**Definition.** An inference rule is "sound" if any clause deduced by applying the rule is a logical consequence of the clauses from which it is deduced.

**Theorem.** Binary resolution is a sound inference rule (1).

To establish the unsatisfiability of a set of clauses, it is sufficient to deduce a contradiction from that set, which is why the property of refutation completeness is relevant. The following definitions characterize the notion of contradiction in the context of clause sets.

**Definition.** A clause with exactly one literal is called a "unit clause" (or simply a "unit").

**Definition.** Two clauses are termed "contradictory unit clauses" if each of the two clauses contains a single literal, if the two are opposite in sign, and if the two literals (ignoring the sign) can be unified. When two such clauses have been obtained, "unit conflict" has been found.

Contradiction can also be defined in terms of the "empty clause," where the empty clause is the empty set of literals. The empty clause can be interpreted as having the value "false." The connection between the two notions of contradiction is established by noting that the resolution of two conflicting units yields the empty clause. Given a set of clauses, a deduction of unit conflict (or of the empty clause) with sound rules of inference completes a proof by contradiction of the unsatisfiability of that set of clauses.

**Definition.** An inference rule (or set of inference rules) is "refutation complete" if there exists a procedure relying on that inference rule (or set of inference rules) solely such that, for every unsatisfiable set of clauses, a proof by contradiction can be obtained using the procedure.

Although binary resolution is a sound inference rule, it is not by itself refutation complete. The following two clauses are a counterexample to its refutation completeness.

$$P(x) \mid P(y) \\ \neg P(x) \mid \neg P(y)$$

All clauses that can be obtained by repeatedly applying binary resolution, starting with the two given clauses, contain exactly two literals. However, if binary resolution is employed together with the inference rule "factoring" (28), the resulting set of inference rules is refutation complete for this example as well as any unsatisfiable set of clauses.

**Definition.** The inference rule "factoring" yields the clause  $C'$  from the clause  $C$  when  $C$  contains two literals  $L1$  and  $L2$  that have the same sign and that can be unified. The clause  $C'$ , called a "factor," is obtained by applying to  $C$  a most general unifier that unifies  $L1$  and  $L2$ . In addition, all factors of factors of  $C$  are themselves factors of  $C$  and are said to be obtained by factoring.

For example, the clause

$$P(f(y)) \mid \neg Q(f(y))$$

is a factor of the clause

$$P(x) \mid P(f(y)) \mid \neg Q(x)$$

which can be seen by unifying the first two literals of the second clause.

**Theorem.** Factoring is a sound inference rule.

**Theorem.** The combination of binary resolution and factoring is refutation complete (1,36).

This theorem might suggest that there exists a decision procedure employing binary resolution and factoring—a procedure that will always correctly identify unsatisfiable and satisfiable sets of clauses. Unfortunately, such is not the case. In fact, no such procedure exists, regardless of which inference rules and strategies are employed. This deep and complex result is based on theorems of Church (see Church's thesis) (37) and Turing (38). In particular, for any given algorithm, finite sets of clauses exist that are satisfiable but for which the algorithm will be unable to establish that fact in a finite amount of time. This lack of a decision procedure is not critical, however, since—at least for proving that some given statement is a theorem—the intent is to prove that some given set of clauses is in fact unsatisfiable. The important considerations for an inference rule (or set of inference rules) are soundness, refutation completeness, and effectiveness.

Although using binary resolution is not adequate for solving most problems, its formulation did lead to the discovery of other closely related and more effective inference rules.

**Definition.** The inference rule "hyperresolution" (18) yields the clause  $C$  by considering simultaneously the clause  $N$  and the clauses  $A_i$ . The clauses  $N$  and  $A_i$  are assumed pairwise to have no variables in common. For hyperresolution to apply, the clause  $N$  must contain at least one negative literal, and the clauses  $A_i$  must each contain only positive literals. If successful, hyperresolution yields a clause  $C$  containing only positive literals. The clause  $C$  is obtained by finding an MGU that (ignoring the sign) simultaneously unifies one positive literal in each of the  $A_i$  with a distinct negative literal in  $N$ , applying the MGU to yield  $N'$  and  $A_i'$  and taking the disjunction of all literals in the  $A_i'$  and in  $N'$  that do not participate in the unification. The clause  $N$  is termed the "nucleus" and the clauses  $A_i$  the "satellites" for that application of hyperresolution. The clause  $C$  is termed a "hyperresolvent."

Hyperresolution applied to the four clauses

1.  $\neg P(x, y) \mid \neg P(y, z) \mid \neg Q(z) \mid R(x, z)$ ,
2.  $P(a, x)$ ,
3.  $P(b, c)$ , and
4.  $Q(c)$

yields

$$R(a, c)$$

as a hyperresolvent by resolving clause 2 with the first literal of clause 1, clause 3 with the second literal, and clause 4 with the third literal.

Where the inference rule of hyperresolution focuses on the signs of the literals to be present in the conclusion, UR resolution (26) focuses on the number of literals to be present. The definition of UR resolution (unit-resulting resolution) can be obtained from the definition of hyperresolution by observing the following constraints. When the satellites are required to be (positive or negative) unit clauses and the nucleus is constrained to have exactly one more literal than the number of satellites and the conclusion is required to be a unit clause, the definition of UR resolution is obtained. Other variations of

binary resolution include unit resolution (24,25), semantic resolution (19), linear resolution (21,22), and lock resolution (23).

None of the cited inference rules applies to that case in which the intention is to treat equality as "built in." To address that case, the inference rule "paramodulation" (20) was formulated.

**Definition.** An "equality literal" is a literal whose predicate is to be interpreted as "equal." The inference rule "paramodulation" yields the clause  $C$  from the clauses  $A$  and  $B$  (that are assumed to have no variables in common) when  $A$  contains a positive equality literal  $K$  and  $B$  contains a term  $t$  that unifies with one of the arguments of  $K$ . Assume without loss of generality that the chosen equality literal  $K$  has the form  $\text{EQUAL}(r, s)$  for terms  $r$  and  $s$ , and that the first argument is that which is being unified with the chosen term  $t$  in  $B$ . The clause  $C$  is obtained from  $A$  and  $B$  with the following procedure. First, find an MGU for the argument  $r$  and the term  $t$ . Second, apply the MGU to  $A$  and  $B$ , yielding  $A', B', K'$ , and  $t'$  as the respective correspondents of  $A, B, K$ , and  $t$ . Third, generate  $B''$  from  $B'$  by replacing  $t'$  by  $s'$ , where  $K'$  is of the form  $\text{EQUAL}(r', s')$ . Finally, form the disjunction of  $B''$  and  $A' - K'$  ( $A'$  minus a single occurrence of  $K'$ ). Clause  $A$  is called the "from clause," clause  $B$  the "into clause," and clause  $C$  a "paramodulant."

For a trivial example, paramodulation applied to the equation  $a + -a = 0$  and the statement " $a + -a$  is congruent to  $b$ " yields in a single step " $0$  is congruent to  $b$ ." In clause form,

$\text{EQUAL}(\text{sum}(a, \text{minus}(a)), 0)$

into

$\text{CONGRUENT}(\text{sum}(a, \text{minus}(a)), b)$

the clause

$\text{CONGRUENT}(0, b)$

is obtained by paramodulation. For a slightly more interesting example, recalling that variables such as  $x$  mean "for all  $x$ ," paramodulation applied to the equation  $a + -a = 0$  and the statement " $x + -a$  is congruent to  $x$ " yields in a single step " $0$  is congruent to  $a$ ." In clause form, from

$\text{EQUAL}(\text{sum}(a, \text{minus}(a)), 0)$

into

$\text{CONGRUENT}(\text{sum}(x, \text{minus}(a)), x)$

the clause

$\text{CONGRUENT}(0, a)$

is obtained. Finally, for a complex and surprising example, paramodulation applied to the equations  $x + -x = 0$  and  $y + (-y + z) = z$  yields in a single step  $y + 0 = -(-y)$ . In clause form, from

$\text{EQUAL}(\text{sum}(x, \text{minus}(x)), 0)$

into

$\text{EQUAL}(\text{sum}(y, \text{sum}(\text{minus}(y), z)), z)$

the clause

$\text{EQUAL}(\text{sum}(y, 0), \text{minus}(\text{minus}(y)))$

is obtained by paramodulation. To see that this last clause is in fact a logical consequence of its two parents, unify the argu-

ment  $\text{sum}(x, \text{minus}(x))$  with the term  $\text{sum}(\text{minus}(y), z)$ , apply the corresponding MGU to both the "from" and "into" clauses and then make the appropriate term replacement.

Where resolution combines in a single step the operations of substitution for variables and cancellation of literals, paramodulation combines in a single step the operations of substitution for variables and replacement of (equal) terms. In contrast to binary resolution, paramodulation operates at the term level rather than at the literal level. Consequently, the use of paramodulation often yields shorter and more natural proofs than does the use of binary resolution. In particular, when a proof calls for the substitution of one term for its equal, the use of binary resolution usually requires several deductions to arrive at the desired conclusion. The use of paramodulation, on the other hand, requires but one deduction to perform the substitution. The difference in the number of deductions is explained in part by the fact that resolution treats each predicate syntactically, in contrast to paramodulation, which treats certain predicates semantically. Specifically, if a predicate is used to mean equality and the appropriate symbols are employed, paramodulation treats that predicate as if it were "understood" ("built in"). In fact, as is especially evident in the third example of how paramodulation works, this inference rule generalizes the usual notion of equality substitution.

Although the use of paramodulation has many advantages, because of operating at the term level rather than at the literal level, uncontrolled application can yield far too many clauses. In particular, if the rule is applied to a pair of clauses where the "into clause" contains many terms, many conclusions may be drawn from that single pair of clauses alone. On the other hand, if applied to that pair, binary resolution will usually yield far fewer conclusions. Simply put, clauses usually contain far fewer literals than terms. Even when all clauses contain only a few terms, uncontrolled application of paramodulation—and, for that matter, of any known inference rule—has the potential of causing a reasoning program to be extremely ineffective. The potential ineffectiveness results from two causes. First, a reasoning program can easily get lost, spending its time focusing on one unwisely chosen clause after another. Second, a reasoning program can easily draw many conclusions that are irrelevant to the question under investigation. What is needed to cope with this potential ineffectiveness is strategy—strategy to direct the reasoning and strategy to restrict the reasoning.

### Strategy

With the formal treatment of inference rules completed, the focus of attention shifts to strategies to control the various inference rules. Although the inference rules of hyperresolution (18), UR resolution (26), and paramodulation (20) each give reasoning programs potentially far more reasoning power than is available with binary resolution, their (uncontrolled) individual or collective use is not effective for solving most problems. In most cases effectiveness is sharply increased with the use of strategy—strategy to direct the reasoning and strategy to restrict the reasoning.

The first strategy—in fact, the notion of strategy in the context of automated theorem proving—was introduced in Ref. 28. That strategy, called "unit preference," directs theorem-proving and reasoning programs to prefer applications of binary resolution to some pair of clauses in which at least one

of the clauses is a unit clause—a clause that contains one literal. The intuitive justification of the value of this approach is the intention of increasing the probability of producing unit clauses. (Resolving two clauses, one of which is a unit clause and one of which is not, produces a resolvent with at least one less literal than the nonunit clause.) Unit clauses in part derive their importance from the fact that almost all proofs by contradiction can be completed by finding two “conflicting units.” (A natural extension of the unit preference strategy can be applied to inference rules other than binary resolution.) Although the introduction in 1964 of the unit preference strategy enabled theorem-proving programs to prove simple theorems that had not previously been proved with such programs, the strategy was discovered to be inadequate for most proof searches. The inadequacy stems from the potential size of the clause set that the program might be forced to search. Even for simple problems, the number of clauses that might be considered can be enormous. Since the unit preference and similar strategies are designed to direct the reasoning, a need also exists for strategies that restrict the reasoning. In general, strategies that restrict the application of inference rules are far more powerful than strategies that direct their application.

In response to the need for restriction strategies, the “set of support strategy” (29) was formulated in 1965. (Although the strategy was originally formulated to restrict the application of binary resolution, it is currently used to effectively control other inference rules employed by automated reasoning programs.) For binary resolution, certain clauses are in effect marked as inadmissible for consideration pairwise by the reasoning program. In particular, the marked clauses are not allowed to act together to deduce additional clauses. Any marked clause can, however, participate in a deduction step if the other clause is not marked. Formally, to employ the set of support strategy, the user chooses a (not necessarily proper) subset  $T$  of the set  $S$  of clauses that represents the theorem to be proved or problem to be solved. For a pair of clauses in  $S$  to be considered for the application of resolution, at least one of the two clauses must be in  $T$ . When a clause deduced with resolution is added to the set of clauses, with respect to the set of support strategy, the clause is treated as if it were an element of the set  $T$ . Such clauses can then be considered with any clause from  $S$  for possible applications of resolution. The pairs of clauses that are not admissible are those pairs  $A$  and  $B$  where both  $A$  and  $B$  are in  $S - T$ .

From one viewpoint, the intent of using the set of support strategy is to block the reasoning program from drawing conclusions that may have nothing to do with the task at hand. From the opposite viewpoint, the object of using the set of support strategy is to cause the reasoning program to focus on certain clauses, using the other clauses merely to complete some deduction step. If the focal clauses are well chosen, the conclusions that are drawn will be relevant to the problem under study. In effect, the set of support strategy permits the user to designate certain clauses as key to the study being made.

For example, if  $S - T$  consists of the basic axioms for the theory from which the problem under study is taken, the program is prohibited from exploring the underlying theory. One effective choice for  $T$  consists of the clauses corresponding to the special hypothesis of the theorem to be proved together with the clauses resulting from assuming the purported theorem false. For the group theory problem cited in the introduction  $T$  would then consist of the clause

$$\text{EQUAL}(\text{prod}(x, x), e)$$

stating that the square of every element is the identity  $e$ , and the clause

$$\neg \text{EQUAL}(\text{prod}(a, b), \text{prod}(b, a))$$

stating that (at least) two elements exist that do not commute. A second effective choice for  $T$  consists of just those clauses resulting from the assumption that the purported theorem is false. For the example from group theory, the clause

$$\neg \text{EQUAL}(\text{prod}(a, b), \text{prod}(b, a))$$

would be the only element of  $T$ . An analysis of either of the two generally described choices for  $T$  shows that the strategy is designed to force a reasoning program to draw conclusions that are relevant to some proof of the theorem under consideration. From a more general viewpoint, recalling that proof by contradiction is the usual goal, the strategy seeks to take advantage of the known consistency of that set of clauses corresponding to the axioms—those clauses that characterize the underlying theory from which the problem has been selected. After all, applying sound inference rules to a consistent set will yield additional consistent information, but a proof of inconsistency is the goal. The set of support strategy is still considered to be the most powerful strategy available to reasoning programs. The strategy is “refutation complete,” as established with the following theorem.

*Set of Support Theorem.* If  $S$  is an unsatisfiable set of clauses, and if  $T$  is a subset of  $S$  such that  $S - T$  is satisfiable, the imposition of the set of support strategy on the combination of binary resolution and factoring preserves the property of refutation completeness for that combination (29).

A third strategy, known as the “weighting strategy” (30), enables the user to direct the reasoning of the computer program according to the user’s knowledge and intuition. The user assigns values to the various concepts and symbols in the problem to be solved, and the program chooses where to focus its attention according to priorities based on those values. The program also purges information from its database when that information has a computed value larger than some preassigned limit for retained information. The weighting strategy has proved very useful and in general more effective than the unit preference strategy for directing the reasoning.

Two other procedures, “subsumption” (1) and “demodulation” (39), are in many cases indispensable when using a reasoning program. With subsumption, the clause  $B$  is discarded in the presence of the clause  $A$  when there exists a substitution for the variables in  $A$  such that, when applied to  $A$ , the result is a subclause of  $B$ . For example, the clause

$$P(x)$$

subsumes the clause

$$P(a)$$

by merely replacing  $x$  by  $a$ . Subsumption can be termed a “deletion” strategy although, when it was formulated, the notion of strategy had not yet been applied to the field of automated theorem proving. Its use purges the database of trivial consequences of existing information. In the cited example, an instance of a clause is subsumed by the clause itself. A more interesting example is provided by the pair of clauses

$$\begin{aligned} P(x) \\ P(a) \mid Q(b) \end{aligned}$$

in which the first clause subsumes the second. Ironically, subsumption acts in opposition to the earlier procedure based on Herbrand instantiation; the former purges instances, and the latter generates them.

Demodulation, on the other hand, rather than removing or adding information to the database, rewrites information by simplifying and normalizing it. [The formulation of demodulation and paramodulation eventually led to the study of complete sets of reductions (40) and rewrite rules.] The corresponding procedure attempts to apply various equality unit clauses, called "demodulators" (39), that have been designated for that purpose. As an example of simplification, the clause

$$\text{EQUAL}(\text{sum}(0, a), b)$$

is immediately rewritten to

$$\text{EQUAL}(a, b)$$

in the presence of

$$\text{EQUAL}(\text{sum}(0, x), x)$$

if the last of the three clauses has been designated a demodulator. As an example of normalization, the clause

$$\text{EQUAL}(\text{prod}(\text{prod}(a, x), b), c)$$

is immediately rewritten to

$$\text{EQUAL}(\text{prod}(a, \text{prod}(x, b)), c)$$

in the presence of

$$\text{EQUAL}(\text{prod}(\text{prod}(x, y), z), \text{prod}(x, \text{prod}(y, z)))$$

if the last of these three has been designated a demodulator.

### Reasoning Programs

The first resolution-based computer program was designed and implemented by Carson (31), and until 1973 it was considered the most powerful theorem-proving program available. The program was designed to prove theorems, most of which were taken from abstract algebra. It relied heavily on the use of strategy. Experiments with Carson's program led to the discovery of the set of support strategy (29) and the concept of demodulation (39). In contrast to that first resolution-based program, which was usable only in batch mode, Allen and Luckham (33) designed and implemented a program that could be used in batch or interactive mode. Their program, also relying on resolution, proved a number of theorems from mathematics. Other useful programs extant in the late 1960s included Green's question-answering program (41), Bledsoe's program (34), and Guard's program (32). Guard's program deserves special mention, for it was the first theorem-proving program permitting interactive use. More important, its use led to finding a new result in mathematics, the lemma from lattice theory known as Sam's lemma. The use of an automated reasoning program to find new results in mathematics did not occur again until 1978 when the program AURA (2) was used to answer a previously open question in algebra (5). AURA was designed and implemented by Overbeek with contributions from Smith, Winker, and Lusk. Among the other programs that were designed and implemented between 1965 and 1985, one particular program, LMA, is of unusual interest even though it is itself not a reasoning program. LMA (3,4), designed and implemented in 1980 by Overbeek, Lusk, and McCune, is used to design automated reasoning programs tai-

lored to the user's specifications. It has been employed to produce the program ITP now used by more than 100 institutions for various types of research and diverse applications.

### Current Applications

The current applications of resolution-based reasoning programs include research in mathematics and formal logic (5-9), design and validation of logic circuits (10,11,42,43), verification of claims made for computer programs, database inquiry, and the reasoning required by various expert systems. A number of successes in these diverse areas have been achieved with AURA, currently (in 1985) considered the most powerful reasoning program available. For example, with its assistance, open questions were solved in ternary Boolean algebra (5), in finite semigroup theory (6), and in equivalential calculus (8,9). Some of the circuits designed (10) by relying on this program are superior (with respect to transistor count) to those previously known. The design of a 16-bit adder was validated (11) with AURA.

Applications for other reasoning programs are also being studied. For example, LMA (3,4) is currently in use as the basis for an expert system to verify certain properties claimed for existing computer programs. ITP, produced from LMA, is being considered for an expert system for car-sequencing problems. The cited examples only hint at the activity, interest, and number of applications for resolution-based reasoning programs.

### Summary

The formulation in 1963 of the inference rule binary resolution by Robinson (1) changed the course of automated theorem proving and, eventually, had a dramatic impact on the use of computers. Briefly, binary resolution is a way of reasoning that considers two statements in the clause language and attempts to deduce some new fact or new relationship. Since the reasoning programs under study before 1963 (13,14) merely substituted terms for the variables in a statement to deduce trivial consequences, Robinson's formulation of resolution represented a promising development. In one sense, that promise was not fulfilled since, for most problems, application of resolution yields too many steps, each representing too small a reasoning step. On the other hand, the promise was more than fulfilled, for the formulation of binary resolution did in fact lead to the discovery of other, more effective inference rules (18-27) and to the discovery of powerful strategies (28-30) to control those rules. For resolution—as well as for all other currently available inference rules—to be effective, strategy to direct and strategy to restrict its application is required.

An accurate evaluation of the full significance of the discoveries that can be directly traced to the formulation of resolution cannot be made until various objectives have been attained. However, one important goal has already been reached. Now (in 1985) computers can be used to assist in the reasoning required in many areas of research and for many applications. To be able to instruct a single computer program to "reason" from given facts and relationships and have it obey the instructions sufficiently well that desired proofs and models and counterexamples are found gives individuals primarily interested in research access to a valuable assistant. Equally, to have that same program reason sufficiently well that important information is provided for solving problems in

design, validation, control, and testing gives individuals primarily interested in some specific application that same advantage. The effectiveness of a computer program that functions as an automated reasoning assistant is proportional to the types of inference rule and strategy it offers and to the excellence of its design and implementation. Excellent resolution-based reasoning programs (2–4) are in fact now being used for various kinds of research and diverse applications.

Whether the use of programs of the type discussed in this entry will continue to spread will be known in a few years. The evidence of the preceding few years already shows that the work begun more than 25 years ago has in fact culminated in success. The previously open questions that were answered (5–9,44), the superior circuits that were designed (10), and the existing circuit designs that were validated (11) are examples of what has been achieved with the assistance of a resolution-based reasoning program. The goal pursued by mathematicians and logicians and (eventually) by various other scientists has been reached. Computer programs are now available that function effectively as automated reasoning assistants.

## BIBLIOGRAPHY

1. J. Robinson, "A machine-oriented logic based on the resolution principle," *JACM* 12, 23–41 (1965).
2. B. Smith, Reference Manual for the Environmental Theorem Prover, An Incarnation of AURA, Technical Report Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1987.
3. E. Lusk, W. McCune, and R. Overbeek, Logic Machine Architecture: Kernel Functions, in D. W. Loveland (ed.), *Lecture Notes in Computer Science*, Vol. 138, Springer-Verlag, New York, pp. 70–84, 1982.
4. E. Lusk, W. McCune, and R. Overbeek, Logic Machine Architecture: Inference Mechanisms, in D. W. Loveland (ed.), *Lecture Notes in Computer Science*, Vol. 138, Springer-Verlag, New York, pp. 85–108, 1982.
5. S. Winker and L. Wos, Automated Generation of Models and Counterexamples and Its Application to Open Questions in Ternary Boolean Algebra, *Proceedings of the Eighth International Symposium on Multiple-Valued Logic*, IEEE, Rosemont, IL, pp. 251–256, 1978.
6. S. Winker, L. Wos, and E. Lusk, "Semigroups, antiautomorphisms, and involutions: A computer solution to an open problem, I," *Math. Computat.* 37, 533–545 (1981).
7. S. Winker, "Generation and verification of finite models and counterexamples using an automated theorem prover answering two open questions," *JACM* 29, 273–284 (1982).
8. L. Wos, S. Winker, R. Veroff, B. Smith, and L. Henschen, "Questions concerning possible shortest single axioms in equivalential calculus: An application of automated theorem proving to infinite domains," *Notre Dame J. Form. Log.* 24, 205–223 (1983).
9. L. Wos, S. Winker, R. Veroff, B. Smith, and L. Henschen, "A new use of an automated reasoning assistant: Open questions in equivalential calculus and the study of infinite domains," *Artif. Intell.* 22, 303–356 (1984).
10. W. Wojciechowski and A. Wojcik, "Automated design of multiple-valued logic circuits by automatic theorem proving techniques," *IEEE Trans. Comput.* C-32, 785–798 (1983).
11. A. Wojcik, Formal Design Verification of Digital Systems, *Proceedings of the Twentieth Design Automation Conference*, Miami Beach, FL, pp. 228–234, 1983.
12. H. Wang, "Towards mechanical mathematics," *IBM J. Res. Devel.* 4, 224–268 (1960).
13. P. Gilmore, "A proof method for quantification theory: Its justification and realization," *IBM J. of Res. Devel.* 4, 28–35 (1960).
14. M. Davis and H. Putnam, "A computing procedure for quantification theory," *JACM* 7, 201–215 (1960).
15. D. Prawitz, H. Prawitz, and N. Voghera, "A mechanical proof procedure and its realization in an electronic computer," *JACM* 7, 102–128 (1960).
16. J. Herbrand, Recherches sur la Théorie de la Démonstration, *Travaux de la Société des Sciences et des Lettres de Varsovie, Classe III Science Mathématique et Physiques*, University of Paris, 1930.
17. J. Herbrand, Investigations in Proof Theory: The Properties of Propositions, in J. van Heijenoort (ed.), *From Frege to Gödel: A Source Book in Mathematical Logic*, Harvard University Press, Cambridge, MA, pp. 525–581, 1967.
18. J. Robinson, "Automatic deduction with hyper-resolution," *Int. J. Comput. Math.* 1, 227–234 (1965).
19. J. Slagle, "Automatic theorem proving with renamable and semantic resolution," *JACM* 14, 687–697 (1967).
20. G. Robinson and L. Wos, Paramodulation and Theorem Proving in First-Order Theories with Equality, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 4, American Elsevier, New York, pp. 135–150, 1969.
21. D. W. Loveland, A Linear Format for Resolution, *Proceedings of the 1968 IRIA Symposium on Automatic Demonstration*, Springer-Verlag, New York, pp. 147–162, 1970.
22. D. Luckham, Refinements in Resolution Theory, *Proceedings of the 1968 IRIA Symposium on Automatic Demonstration*, Springer-Verlag, New York, pp. 163–190, 1970.
23. R. Boyer, Locking: A Restriction on Resolution, Ph.D. Thesis, University of Texas, Austin, TX, 1971.
24. D. Kuehner, Some Special Purpose Resolution Systems, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 7, American Elsevier, New York, pp. 117–128, 1972.
25. L. Henschen and L. Wos, "Unit refutations and Horn sets," *JACM* 21, 590–605 (1974).
26. J. McCharen, R. Overbeek, and L. Wos, "Problems and experiments for and with automated theorem proving programs," *IEEE Trans. Comput.* C-25, 773–782 (1976).
27. L. Wos, R. Veroff, B. Smith, and W. McCune, The Linked Inference Principle, II: The User's Viewpoint, in R. E. Shostak (ed.), *Lecture Notes in Computer Science*, Vol. 170, Springer-Verlag, New York, pp. 316–332, 1984.
28. L. Wos, D. Carson, and G. Robinson, The Unit Preference Strategy in Theorem Proving, *Proceedings of the Fall Joint Computer Conference, 1964*, Thompson Book Co., New York, pp. 615–621, 1964.
29. L. Wos, D. Carson, and G. Robinson, "Efficiency and completeness of the set of support strategy in theorem proving," *JACM* 12, 536–541 (1965).
30. J. McCharen, R. Overbeek, and L. Wos, "Complexity and related enhancements for automated theorem-proving programs," *Comput. Math. Appl.* 2, 1–16 (1976).
31. L. Wos, G. Robinson, and D. Carson, Some Theorem-Proving Strategies and Their Implementation, Technical Memo 72, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1964.
32. J. Guard, F. Oglesby, J. Bennett, and L. Settle, "Semi-automated mathematics," *JACM* 16, 49–62 (1969).
33. J. Allen and D. Luckham, An Interactive Theorem-Proving Program, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 5, American Elsevier, New York, pp. 321–336, 1979.
34. W. Bledsoe and P. Bruell, "A man-machine theorem-proving system," *Artif. Intell.* 5, 51–72 (1974).
35. L. Wos, R. Overbeek, E. Lusk, and J. Boyle, *Automated Reasoning: Introduction and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

36. C. Chang and R. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.
37. A. Church, "An unsolvable problem of number theory," *Am. J. Math.* **58**, 345-363 (1936).
38. A. Turing, "On computable numbers, with an application to the Entscheidungs problem," *Proc. Lond. Math. Soc.* **42**, 230-265 (1936).
39. L. Wos, G. Robinson, D. Carson, and L. Shalla, "The concept of demodulation in theorem proving," *JACM* **14**, 698-704 (1967).
40. D. Knuth and P. Bendix, Simple Word Problems in Universal Algebras, in J. Leech (ed.), *Computational Problems in Abstract Algebra*, Pergamon, New York, pp. 263-297, 1970.
41. C. Green, Theorem Proving by Resolution as a Basis for Question-Answering Systems, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 4, American Elsevier, New York, pp. 183-205, 1969.
42. W. Wojciechowski and A. Wojcik, Multiple-Valued Logic Design by Theorem Proving, *Proceedings of the Ninth International Symposium on Multiple-Valued Logic*, IEEE, New York, pp. 196-199, 1979.
43. W. Kabat and A. Wojcik, Automated Synthesis of Combinational Logic using Theorem Proving Techniques, *Proceedings of the Twelfth International Symposium on Multiple-Valued Logic*, IEEE, New York, pp. 178-199, 1982.
44. L. Wos and S. Winker, Open Questions Solved with the Assistance of AURA, in W. W. Bledsoe and D. Loveland (eds.), *Automated Theorem Proving: After 25 Years*, American Mathematical Society, Providence, RI, pp. 73-88, 1984.

### General References

- R. Boyer and J. Moore, *A Computational Logic*. Academic, New York, 1979.
- R. Boyer and J. Moore, Proof Checking the RSA Public Key Encryption Algorithm, Technical Report 33, The Institute for Computing Science, University of Texas, Austin, TX, 1982.
- D. Loveland, *Automated Theorem Proving: A Logical Basis*, North-Holland, New York, 1970.

L. WOS  
Argonne National Laboratory

R. VEROFF  
University of New Mexico

This work was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract W-31-109-Eng-38.

## ROBOT-CONTROL SYSTEMS

The practice of automatic control has its origins in antiquity (1,2). It is only recently—within the middle decades of this century—that a body of scientific theory has been developed to inform and improve that practice. Control theorists tend to divide their history into two periods. A "classical" period prior to the sixties witnessed the systematization of feedback techniques based on frequency-domain analysis dominated by applications to electronics and telephony. A "modern" period in the sixties and seventies was characterized by a growing concern with formal analytical techniques pursued within the time domain motivated by the more stringent constraints posed by space applications and the enhanced processing capa-

bility of digital technology (3). The hallmark of control theory has been, by and large, a systematic exploitation of the properties of linear dynamic systems whether in the frequency or time domain. Its great success in applications is a remarkable tribute to the diverse range of physical phenomena for which such models are appropriate.

The field of robotics (qv) presents control theorists with a fascinating and novel domain. Although numerically controlled kinematic chains with a few degrees of freedom have been available for over two decades, it is only within the last five years that mechanical systems with many degrees of freedom, each independently actuated, have been wedded to dedicated computational resources of considerable sophistication. To begin with, the dynamic behavior of such systems appears to depart dramatically from the familiar linear case. Further, typical robotic tasks involve complex interactions with diverse environments possessing, of themselves, kinematics and dynamics, which may change abruptly throughout the course of desired operations. Finally, the complexity of these tasks makes even their specification problematic for purposes of control.

This entry attempts simultaneously to provide general readers with an introductory account of the nature of control theory and its application to robotics, and offers control and robot theorists a brief (and necessarily incomplete) look at the contemporary research horizons. For concreteness, the discussion is limited to robot arms—open kinematic chains with rigid links. For reasons of brevity, much of the tutorial discussion is concerned with robot motion alone, even though general manipulation tasks are the more interesting application. The first section, an elementary introduction to the problems and methods of general control theory, introduces some fundamental properties of dynamic systems, the nature of stabilizing feedback structures, and the capabilities of the servomechanisms that result, all in the context of a very rudimentary "robot arm"—the simple pendulum. Hopefully, a general college physics course and a semester's introduction to differential equations will be background enough to gain some appreciation of the material.

The tone and mathematical sophistication of this entry shifts in the sequel in favor of a more specialized reader. General Robot-Arm Dynamics presents a brief review of Lagrange's equations, leading to the definition of the general rigid-body model of robot dynamics, and an equally brief examination of its more significant omissions. Feedback Control of General Robot Arms concerns the application of general feedback techniques to the problem of robot task encoding and control. This treatment departs from traditional control theory by paying as much attention to issues of task definition and encoding as to control algorithms. The intent is to suggest how relatively simple error-driven control structures can afford the command of a surprisingly rich array of robot tasks. Servo Control of General Robot Arms is more in the character of a survey of the contemporary robot-arm-control literature. It will be observed that this literature returns to the traditional control theoretic task-encoding paradigm of the servomechanism. Unlike any of the previous sections, the discussions to be found in these last two sections are incomplete not primarily in consequence of limited space but because the relevant theory has not yet been developed. Some attempt will be made to point out the important gaps.

Throughout the entry technical terms of general impor-



tance for the exposition will be italicized, given a reference for more detail reading, as well as defined in subsequent text if the meaning is not clear from the context. Other technical terminology that may be current in the literature will be simply placed within quotation marks and used colloquially. More idiosyncratic terminology and notation is defined in the appendix.

### Single-Degree-of-Freedom Robot Arm

This section provides an introduction to the established body of control theory as developed during the "classical" period, yet expounded in the "modern" language (referring to the historical periods introduced above). The "frequency-domain" techniques of classical control theory lie at the foundation of the discipline and offer design methods proven over the last 60 years in the context of a great variety of physical problems. Unfortunately, the class of dynamic systems represented by high-performance robots with revolute arms is not amenable to a general rigorous analysis using these tools: strongly coupled nonlinear dynamics do not admit representation by transfer function. Some researchers, e.g., Horowitz (4), have successfully used modified frequency-domain methods for restricted nonlinear control problems. For the purposes of a general tutorial, the central insights of control theory may be translated quite readily into the more widely familiar elementary methods of ordinary differential equations. Fortunately, similar (although rather more advanced) methods afford the translation of some of these into the general robotic domain as well, as will be demonstrated in the last two sections. Thus, by avoiding the language of transfer functions, we seek to reach a broader audience in this section, while motivating the more technical discussion in future sections.

In this section attention is limited to the case of a single-degree-of-freedom mechanical control system—the actuated simple pendulum. Although this system cannot convey the scope of 60 years of control research, the insights motivated by such second-order linear-time-invariant systems pervade the field. At the same time this system represents the simplest possible revolute robot arm.

The notion of a dynamical model is introduced below, as is the need for control theory, hopefully making clear that the fundamental problem of control is not a consequence of limited power but of limited information. Constraints of space have unfortunately precluded the addition of sections concerning many "modern" techniques such as optimal control, stochastic filtering, or learning theory as applied to this system. The reader may note that each subsection here anticipates a more specialized treatment of analogous material for the general robot arm in later sections.

**Dynamics: A Source of Delay and Uncertainty.** A simple pendulum consists of a mass  $M$  attached via a rigid (massless) link to a joint that permits rotational motion limited to the plane on which the link lies. Perfect angular position and velocity sensors located at the joint deliver exact measurements,  $\theta$ ,  $\dot{\theta}$ , respectively, continuously and instantaneously. A perfect actuator has been placed at the joint: this idealized device has no power limitations and hence can deliver arbitrarily large torques,  $\tau$ , instantaneously. We assume that the plane of motion is horizontal so that there are no gravitational or other disturbance torques. Such artificial assumptions are relaxed

very soon and serve here merely to underscore the insight that the fundamental problems of control arise from uncertain information and intrinsic delay rather than power constraints, as discussed above.

The control problem may be rendered roughly as follows: design an algorithm that produces a time profile of torques,  $\tau(t)$ , so as to elicit some specified behavior of the simple pendulum. In the context of robotics, the following terminology (which is alien to conventional control theory) proves quite useful. The precise nature of the desired property determines what might be called the *task domain*, and the particular instance, an *encoded task specification*, or *plan*.

**Newtonian Dynamic Model.** In order to think about control algorithms, we first require some understanding of the relationship between adjustments in  $\tau$  and resulting changes in  $\theta$ . This relationship is completely specified by Newton's law relating torque to angular acceleration:

$$M\ddot{\theta} = \tau$$

This is a system with *memory*: changes in  $\theta$  (and, hence,  $\theta$ , itself) at any time  $t$  depend on the past history of  $\tau$  rather than simply on its value at time  $t$ . The fact that physical systems give rise to dynamic rather than memoryless relationships necessitates the need for a theory of control, as will be shown directly.

For the purposes of this entry it will prove more convenient to express such relationships, second-order differential equations involving  $n$  variables, in the equivalent form of first-order differential equations involving  $2n$  variables. Defining

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \triangleq \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

to be the *state variable* expressed in *phase space* emphasizes the fact obtaining from elementary properties of differential equations that the future behavior of the system,  $x(t)$ ,  $t > t_0$ , is entirely determined by its initial conditions,  $x_{10} \triangleq \theta(t_0)$ ,  $x_{20} \triangleq \dot{\theta}(t_0)$ , and future values of the *control input*, i.e., the torque,  $u(t) \triangleq \tau(t)$ ,  $t > t_0$ . For *time-invariant systems* such as this, the behavior is independent of initial time, and it will be assumed in the sequel that  $t_0 \triangleq 0$ . These definitions are more carefully discussed in standard control texts (5–7).

The system may now be specified by phase-space dynamics of the form

$$\dot{x} = f(x, u)$$

with the *vector field* (8) given as

$$f(x, u) = A_0 x + b u \quad (1)$$

where

$$A_0 \triangleq \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad b \triangleq \begin{bmatrix} 0 \\ 1/M \end{bmatrix} u$$

According to the description of the system given above, the entire state variable is measured; thus, the system *output*  $y$  takes the form

$$y(t) = x(t)$$

A traditional circumstance of linear control theory is that the system output—the set of available measurements—contains incomplete information from which it is required to reconstruct the entire state. In the context of robotics this situation is most typically reversed: the system state (joint positions

and velocities) are available, whereas the output (work space positions, velocities, and forces) cannot easily be directly measured.

**Need for a Theory of Control.** In the task domain of *set-point regulation* consider the following task specification: bring  $\theta$  to some desired position  $\theta_d$  and keep it there. Given the ideal context described above, it is quite easy to come up with ad hoc control algorithms.

A *unit impulse* is modeled by the "Dirac delta function,"  $\delta_t(\zeta)$ , defined by

$$\int_{-\infty}^{\infty} f(\zeta)\delta_t(\zeta) d\zeta = f(t)$$

an infinitesimally rapid amount of infinitely large magnitude possessed of unit area (finite energy) that our ideal actuator is able to deliver. An obvious procedure that requires no great body of theory might be the following. Measure the present position and velocity,  $x(0) = [x_{10}, x_{20}]^T$ , and apply an impulse torque at the same instant,

$$u_{\text{start}}(t) \triangleq M(1 - x_{20})\delta_0(t)$$

An impulse has the effect of resetting initial conditions; thus Eq. 1 implies that a new constant angular velocity results,

$$x(t) = \begin{bmatrix} t + x_{10} \\ 1 \end{bmatrix} \quad t > 0$$

and the desired position,  $x_1(t) = \theta_d$ , is achieved at time  $t^* \triangleq \theta_d - x_{10}$ . If, at this instant, a second impulse,

$$u_{\text{stop}} \triangleq -M\delta_{t^*}(t)$$

is applied in the opposite direction of motion, the new velocity is canceled exactly, and the end effector comes to rest in the desired position and remains there for all  $t > t^*$ . The algorithm,

$$u = u_{\text{start}}(t) + u_{\text{stop}}(t)$$

requires a priori knowledge of  $M$ ; instantaneous measurement of and actuation energy exactly proportional to  $x(0)$ ; and an exact timer for marking  $t^*$ . Note that despite our best efforts to idealize the capabilities of sensors and actuator, some finite time must necessarily elapse between the application of the control torque and the desired result. This, too, could be obviated by the further idealization of applying a "unit doublet"—the derivative of the delta function. Such a degree of unreality is not introduced here because, unlike the impulse, the doublet cannot be even approximated by real actuators.

Now suppose that the prior estimate of  $M$ ,  $\hat{M}$ , has some error (e.g., suppose the robot is holding an object whose mass is not known a priori). If the same control is applied, substituting  $\hat{M}$  for  $M$ , the true response of the system will be

$$x(t) = \begin{bmatrix} \left[ \frac{\hat{M}}{M} + \left(1 - \frac{\hat{M}}{M}\right)x_{20} \right] t^* + \left(1 - \frac{\hat{M}}{M}\right)x_{20}t \\ \left(1 - \frac{\hat{M}}{M}\right)x_{20} \end{bmatrix} \quad t > t^*$$

Thus, finite and increasing error results from arbitrarily small inaccuracy in  $\hat{M}$ . It is easy to see that the same problems would result given any inaccuracy in the sensors or magnitude of energy delivered by the actuator. Certainly, a subsequent check of true response at some future time,  $x(t^* + \varepsilon)$ , would reveal whether or not such errors had occurred, and a similar course of action could be planned based on the new observation. But there is no systematic procedure for performing such

checks and readjustments: a "higher level" of control authority would be required to decide when they should be effected. More disturbing, it is not yet clear that any systematic application and reapplication of this procedure exists that can guarantee subsequent improvement from one response to the next. This sort of error propagation is characteristic of *unstable* systems.

The origins of control theory, then, rest in the following observations. Dynamic systems give rise to delay that must be taken into account by any control strategy regardless of available actuator power and sensor accuracy. Moreover, information regarding the real world is inevitably uncertain and may have adverse effect on performance no matter how small the uncertainty or powerful and accurate the apparatus.

**Feedback Control: Behavior of Error-Driven Systems.** The difficulties described above suggest the desirability of control strategies that make systematic and continual use of actuator information in order to reduce successively the performance errors caused by initial uncertainty. This study—the discovery and elucidation of feedback algorithms—is arguably the most profound contribution of control theory to physical science. Here we present some of the basic results from a purely control-theoretic perspective. Below, suitable generalizations begin to suggest a unified approach to dynamically sound robot task-encoding methodologies.

A feedback algorithm is essentially an error-driven control law. Presented with a linear system, it makes sense to investigate feedback laws that are linear in the errors as well. In the context of set-point regulation, the errors in question are the distance of the true angular position from the desired and the true angular velocity from zero. The most general linear function of these two errors is

$$u \triangleq \gamma_1(\theta_d - x_1) + \gamma_2(0 - x_2) \quad (2)$$

defining a class of algorithms known as *PD* ("proportional and derivative") control schemes. According to our model of system dynamics (Eq. 1), the resulting *closed-loop* system is a homogeneous linear-time-invariant differential equation,

$$\dot{y} = A_1 y \triangleq \begin{bmatrix} 0 & 1 \\ \frac{-\gamma_1}{M} & \frac{-\gamma_2}{M} \end{bmatrix} y \quad (3)$$

in the translated coordinate system,

$$y \triangleq \begin{bmatrix} x_1 - \theta_d \\ x_2 \end{bmatrix}$$

The desired end position is an *equilibrium state* of this closed-loop system; i.e., if the initial position and velocity of the arm were exactly at  $(\theta_d, 0)$  to begin with, the resulting future trajectory would remain there for all time. An equilibrium state of a dynamical system that has the property that solutions originating sufficiently near remain near and asymptotically approach it in the future is called *asymptotically stable*. All those initial conditions that are near enough to asymptotically approach an asymptotically stable equilibrium state are said to lie within its *domain of attraction* (9).

**Stability of Closed-Loop System.** We seek to show that this algorithm produces an asymptotically stable closed-loop equilibrium state (the desired end point) whose domain of attraction includes all positions and velocities. This desirable property may be shown to hold in a number of ways: the following demonstration is the only means that may be rigorously ex-

tended to the general case of revolute arms with many degrees of freedom, as shown below.

It may be observed that the algorithm and consequent closed-loop dynamics would be the intrinsic result of introducing a physical spring, with constant  $\gamma_1$ , stretched between the desired and true position, along with a viscous damping mechanism opposing motion with force proportional to velocity, with constant  $\gamma_2$ . Accordingly, considerable insight into the asymptotic behavior of the resulting system may be obtained by studying its mechanical energy.

This is defined as the sum,  $v = \kappa + \mu$  of kinetic energy, due to the velocity of the mass,

$$\kappa \triangleq \frac{1}{2} M \dot{y}_2^2$$

and potential energy stored in the spring,

$$\mu \triangleq \frac{1}{2} \gamma_1 y_1^2$$

The change in energy of the closed-loop system is expressed as

$$\begin{aligned} \dot{v} &= \gamma_1 y_1 \dot{x}_1 + M y_2 \dot{x}_2 \\ &= \gamma_1 y_1 y_2 - \gamma_1 y_1 y_2 - \gamma_2 y_2^2 \\ &= -\gamma_2 y_2^2 \leq 0 \end{aligned} \quad (4)$$

hence, if  $\gamma_2 > 0$ ,  $v$  must decrease whenever the velocity is not zero. If  $M, \gamma_1$  are positive as well,  $v$  is positive, except at the desired state,  $y = 0$ , where  $\theta = \theta_d, \dot{\theta} = 0$ . It is intuitively clear (and will be rigorously demonstrated in greater generality within the proof of Theorem 1 below) that these conditions guarantee  $v$  will tend asymptotically toward zero and, hence, that  $y(t)$  approaches zero as well.

This argument employs the total energy as a *Lyapunov function* (9). It demonstrates that the control algorithm succeeds, asymptotically, in accomplishing the desired task: *stabilizing*

the system with respect to  $\begin{bmatrix} \theta_d \\ 0 \end{bmatrix}$  solves the set-point regulation problem for that end point. It has been already remarked that this stability property is *global* in the sense that any initial position and velocity of the robot arm will "decay" toward the desired equilibrium state. Moreover, no exact information regarding the particular value of  $M$  has been used to achieve the result other than the assumption that it is positive. Since the choice of  $\theta_d$  was arbitrary, it is also clear that the analogous feedback algorithm will stabilize the system around any other desired zero velocity state,  $\begin{bmatrix} \theta'_d \\ 0 \end{bmatrix}$ , as

well, with no further readjustment of the "feedback gains,"  $\gamma_1, \gamma_2$ . The PD controller provides a general solution to the set-point regulation problem.

**Robust Properties of Stable Systems.** The model proposed for the single-degree-of-freedom arm (Eq. 1) cannot be exactly accurate: there will be inevitable small disturbance forces and torques placed on the shaft and arm varying in position and over time. Moreover, real actuators, even those possessed of ample power, are subject to imprecision in the profile of torques or forces output in response to any command. If the cumulative effect of all these uncertainties is small, Eq. 3 may be more accurately written in the form

$$\dot{y} = A_1 y + b \varepsilon(y, t)$$

where the scalar "noise" function is bounded,  $|\varepsilon(y, t)| < \varepsilon_0$ , by some small  $\varepsilon_0 > 0$ . Another advantage of the feedback-control algorithm developed above is that the inevitably resulting errors in performance remain strictly smaller than  $\varepsilon_0$ , no matter

what the form of the noise function,  $\varepsilon(y, t)$ . This, again, may be demonstrated in a variety of ways: in keeping with the philosophy of exposition detailed in the beginning of this section, we appeal to a modified Lyapunov analysis.

Define the symmetric matrix

$$P \triangleq \begin{bmatrix} \gamma_1 & \frac{1}{2} \gamma_2 \\ \frac{1}{2} \gamma_2 & M \end{bmatrix}$$

and define the modified Lyapunov candidate,

$$v \triangleq \frac{1}{2} y^T P y = \frac{1}{2} [\gamma_1 y_1^2 + \gamma_2 y_1 y_2 + M y_2^2]$$

The analytical results below depend on the fact that this is a *positive definite* function—i.e.,  $y^T P y \geq 0$  with equality if  $y = 0$ . For this, it is necessary and sufficient that  $P$  be a positive definite matrix—i.e.,  $\gamma_1, M > 0$  and

$$M > \frac{\gamma_2^2}{4\gamma_1} \quad (5)$$

which condition requires at least a known lower bound on  $M$  to be verified. A slightly more complicated choice of  $P$  may be found yielding the same result with no additional information requirement concerning  $M$ . The present analysis is intended partially to anticipate the discussion of transient performance in the next section, where relative magnitudes of all the parameters become important. Assuming this additional information is available and the conditions of Eq. 5 are met, we have

$$\begin{aligned} \dot{v} &= y^T P A_1 y + y^T P b \varepsilon \\ &= -y^T P y \frac{\gamma_2}{2M} + y^T P b \varepsilon \end{aligned}$$

since  $y^T P A_1 y = -y^T P y (\gamma_2/2M)$ , and, completing the square,

$$\begin{aligned} y^T P A_1 y &= -\frac{\gamma_2}{4M} \left[ y^T P y - \frac{4M^2}{\gamma_2^2} b^T P^{-1} b \varepsilon^2 \right] \\ &\quad - \frac{\gamma_2}{4M} \left[ y - \frac{2M}{\gamma_2} P^{-1} b \varepsilon \right]^T P \left[ y - \frac{2M}{\gamma_2} P^{-1} b \varepsilon \right] \\ &\leq -\frac{\gamma_2}{4M} \left[ y^T P y - \frac{4\varepsilon_0^2 \gamma_1}{\gamma_2^2 (4M\gamma_1 - \gamma_2^2)} \right] \end{aligned}$$

Thus,  $v$ , and, hence,  $\|y\|$ , decrease whenever

$$\|y\|^2 > \frac{4\varepsilon_0^2 \gamma_1}{\gamma_2^2 (4M\gamma_1 - \gamma_2^2) \|P\|^2}$$

This implies that  $\|y\|$  decays asymptotically to a magnitude bounded, at most, by the constant on the right-hand side of the previous inequality. Thus, with no regard to the particular form of the disturbance,  $\varepsilon$ , other than a fixed upper bound on its magnitude,  $\varepsilon_0$ , the steady-state error resulting from the PD feedback algorithm can be arbitrarily small by increasing the gains.

**Adjustment of Transient Response: High-Gain Feedback and Pole Placement.** That the resort to stabilizing state feedback results in a system with desirable "steady-state" properties was discussed under Stability of Closed-Loop System. In the previous section that feedback algorithm is seen to reduce the sensitivity of the limiting behavior to unmodeled disturbances,  $\varepsilon$ , and this benefit is enhanced by increasing the magnitude of the gains. Recall, however, that the latter result involved an additional assumption (Eq. 5) regarding their relative magnitudes in comparison to  $M$ . This section presents a more general argument for the benefits of what may be termed "high-gain feedback"—the choice of gains with the largest

possible magnitude consistent with the capabilities of actuators along with the satisfaction of conditions on their relative magnitude to be developed below. In contrast to such feedback strategies, if more information is available regarding the parameters of the system to be controlled (in this case knowledge of  $M$ ), the resulting transient behavior may be shaped much more precisely by the technique of *pole placement*.

A glance at Eq. 4 shows that the magnitude  $\gamma_2$  governs the rate of decay of  $v$ , suggesting, in the first instance, that the rate of convergence to the desired position may be increased by adjusting that constant. Unfortunately, the rate of decay of  $v$  is governed as well by  $\gamma_2$ , and this results in a more complicated situation. The effects of increasing  $\gamma_2$  indiscriminately by measuring its relative magnitude according to a parameterization by the positive scalar,  $\zeta$ , is expressed as

$$\gamma_2 = 2\zeta[\gamma_1 M]^{1/2}$$

usually called the "damping coefficient" in the systems literature (3)]. Define a direction in the translated phase plane (i.e.,  $y$  coordinates),

$$c \triangleq \begin{bmatrix} \zeta + (\zeta^2 - 1)^{1/2} \\ \left[\frac{M}{\gamma_1}\right]^{1/2} \end{bmatrix}$$

(this is the eigenvector of  $A_1^T$  corresponding to the smaller of its two eigenvalues), and by studying the projection onto the unit vector in this direction,  $\hat{c} \triangleq c/\|c\|^2$ , note that the rate of decrease of this component along the response trajectories of the closed-loop system is

$$\begin{aligned} \frac{d}{dt}[y^T \hat{c}]^2 &= 2y^T \hat{c} \dot{\hat{c}}^T y \\ &= 2y^T \hat{c} \dot{\hat{c}}^T A_1 y \\ &= 2y^T \hat{c} \dot{\hat{c}}^T y \lambda \\ &= \lambda 2[y^T \hat{c}]^2 \end{aligned}$$

where

$$\lambda = -\left[\frac{M}{\gamma_1}\right]^{1/2} (\zeta - (\zeta^2 - 1)^{1/2})$$

(this is the smaller of the two eigenvalues of  $A_1^T$ ). For fixed values of  $\gamma_1$ ,  $M$ ,  $\lambda$  approaches zero as  $\zeta$  increases. Thus, the same initial position will converge to the desired position at a slower and slower rate as  $\gamma_2$  is made much larger than  $\gamma_1$ . By guaranteeing that  $\gamma_1$  grows at least as the square of  $\gamma_2$ , this possibility is ruled out. Note that the modified Lyapunov analysis of the previous section depends on this assumption, as does the robust tracking result discussed below.

On the other hand, significant problems result from choosing too large a magnitude of  $\gamma_1$  relative to  $\gamma_2$ , as may be seen in standard control texts (3,10,11). However, if  $\gamma_1$  is chosen roughly proportional to  $\gamma_2^2$ , it is generally safe to predict that better transient performance will result from increased gains without the explicit knowledge of  $M$  required to design the feedback algorithm (Eq. 2). It should be noted that the presence of higher order dynamics than modeled in Eq. 1 will virtually guarantee that gain increases past a certain magnitude result in deleterious performance and, ultimately, destabilize the closed-loop system. Although the desired position is never attained exactly in finite time, the time required to reach arbitrarily small neighborhoods of the desired position from a given distance away may be made arbitrarily short by increasing the magnitude of the gains. Of course, in practice,

given the inevitable power constraints of the real world there will be some upper limit on the magnitude of  $\gamma_1$ ,  $\gamma_2$  that may be implemented and, hence, on the rate of convergence toward the desired goal that may be attained.

The transient characteristics of a closed-loop system resulting from linear state feedback (Eq. 2) are completely determined by the *poles*—eigenvalues of the resulting system matrix,  $A_1$  in Eq. 3—and, hence, by the roots of a second-order polynomial whose coefficients are exactly specified by the second row of that array (3,10,11). Thus, concern regarding transient performance may be precisely addressed only if full information regarding the parameters of the system is available. Namely, if  $M$  is known, the feedback strategy

$$u = M(k_1 \theta + k_2 \dot{\theta}) \quad (6)$$

results in a closed-loop system with vector field

$$A_2 x \triangleq \begin{bmatrix} 0 & 1 \\ -k_1 & -k_2 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$$

whose eigenvalues are exactly

$$-\frac{1}{2}(k_2 \pm [k_2^2 - 4k_1]^{1/2})$$

**Servo Problem.** The most general task domain of classical control theory is the servo problem. Task specification is by means of a *reference trajectory*—a time-varying signal,  $x_d(t)$ , which it is desired that the system output,  $x(t)$ , should repeat, or *track*, as closely as possible. Typically, the control law takes the form

$$u_{\text{serv}} = u_{\text{fb}} + u_{\text{pc}}$$

where  $u_{\text{fb}}$  is some stabilizing feedback algorithm, e.g., Eq. 2 or 6, and

$$u_{\text{pc}} \triangleq \Gamma[x_d]$$

is some appropriately conditioned or *precompensated* form of the reference signal. A thorough treatment of these ideas can be found in standard classical texts (3,10,11).

**Forced Response of Linear Systems.** Linear-time-invariant systems constitute an important exception to the general rule that the output of a forced dynamic system has no closed-form expression involving elementary functions. Consider a general example of such a system,

$$\dot{x} = Ax + bu \quad (7)$$

Let the initial position and velocity be specified by  $x = [\theta_d, \dot{\theta}_d]^T$ , and define

$$\exp\{tA\} \triangleq \sum_{k=0}^{\infty} (tA)^k/k!$$

Recall that this sum converges for all matrices  $A$  and real values  $t$  and that its (operator) norm decays with increasing values of  $t$  if the eigenvalues of  $A$  have negative real part. It may be verified by direct computation that

$$x(t) = \exp\{tA\}[x_0 + \int_0^t \exp\{-\tau A\}bu'(\tau) d\tau] \quad (8)$$

satisfies the closed-loop dynamic equations (Eq. 3). The availability of this exact "input/output description" is the basis for the powerful frequency-domain methods of classical control theory (3,10,11). As has been stated in the introduction, these techniques are deliberately avoided throughout this section

since they have no rigorously founded counterparts in the context of the general nonlinear robot dynamics to be discussed in the sequel.

Nevertheless, given a specified trajectory,  $x_d(t) = [\theta_d(t), \dot{\theta}_d(t)]^T$ , it is now possible to provide a complete account of the efficacy of any particular control input by examining the resulting error,

$$e(t) \triangleq x_d(t) - x(t)$$

Assume a control of the form  $u \triangleq u_{fb} + u_{pc}$ , as above. The closed-loop plant equations are now of the form of Eq. 7,

$$\dot{x} = A_1 x + b u_{pc} \quad (9)$$

with  $A = A_1$ , and  $\dot{u} = u_{pc} = \Gamma[x_d]$ . (Since the gains  $\gamma_1, \gamma_2$  have been chosen strictly greater than zero to stabilize the desired equilibrium position, there is no question as to the invertibility of  $A_1$ .) Integrating the second term in  $x(t)$  of Eq. 8 by parts twice affords the expression

$$e(t) = x_d(t) - A_1^{-2}[\exp\{tA_1\}a_0 - A_1 b u_{pc}(t) - b \dot{u}_{pc}(t) - \int_0^t \exp\{(t-\tau)A_1\} b \ddot{u}_{pc}(\tau) d\tau] \quad (10)$$

where we define for convenience the controlled "initial acceleration,"

$$a_0 \triangleq A_1^2 x(0) + A_1 b u_{pc}(0) + b \dot{u}_{pc}(0)$$

**Inverse Dynamics.** If it is desired that the true response of the closed-loop system track the reference signal exactly, the most obvious recourse is to "inverse dynamics." Suppose  $M$  is exactly known, and  $u_{fb}$  has been chosen in the form of Eq. 6 to achieve some specified set of poles. Assume that the output of the stabilized system Eq. 3 is exactly the desired signal,  $\theta(t) \equiv \theta_d(t)$ : it follows that all derivatives are equivalent as well,  $\dot{\theta} \equiv \dot{\theta}_d$ ,  $\ddot{\theta} \equiv \ddot{\theta}_d$ , and, hence, solving for  $u$  in the second line of Eq. 9, that

$$u_{id}(t) \equiv M(\ddot{\theta}_d + k^T \begin{bmatrix} \theta_d \\ \dot{\theta}_d \end{bmatrix})$$

In terms of the framework above, this corresponds to a choice for  $\Gamma$  of the form

$$\Gamma_{id}[x_d] \triangleq \frac{1}{b^T b} b^T [\dot{x}_d - A_2 x_d]$$

Using frequency-domain analysis, it is easy to see that this control input is a copy of the reference signal fed through dynamics whose transfer function is the reciprocal of the feedback-stabilized plant. From the point of view of time-domain analysis, the resulting closed-loop system expressed in error coordinates takes the form

$$\begin{aligned} \dot{e} &= \dot{x}_d - A_2 x - \frac{1}{b^T b} b b^T [\dot{x}_d - A_2 x_d] \\ &= A_2 e + [I - \frac{1}{b^T b} b b^T] [\dot{x}_d - A_2 x_d] \\ &= A_2 e \end{aligned}$$

since  $[I - 1/b^T b b b^T]$  is a projection onto the subspace of  $\mathbb{R}^2$  orthogonal to  $b$ , and  $\dot{x}_d - A_2 x_d$  lies entirely in the image of  $b$ . Thus, appealing to a Lyapunov analysis once more, if  $v \triangleq \frac{1}{2} e_2^2 + \frac{1}{2} k_1 e_1^2$ ,  $\dot{v} = -k_2 e_2^2$ , which implies that the error is nonincreasing and from which it can be deduced as well that (see Theorem 1)  $e$  tends toward zero asymptotically. For purposes of comparison

it is worth displaying the actual form of the closed-loop error, [which may be computed from the exact I/O (Eq. 10)]

$$\begin{aligned} e(t) &= -A_1^{-2}[\exp\{tA_1\}a_0 - \dot{x}_d] \\ &\quad + \int_0^t \exp\{(t-\tau)A_1\} b \ddot{u}_{id}(\tau) d\tau \\ &\quad (\text{since } \ddot{x}_d = A_2^2 x_d + A_2 b u_{id} + b \ddot{u}_{id}) \\ &= -A_1^{-2} \exp\{tA_1\} [a_0 - \dot{x}_d(0)] \\ &\quad + \int_0^t \frac{d}{d\tau} \exp\{-\tau A_1\} \dot{x}_d(\tau) d\tau \\ &\quad + \int_0^t \exp\{-\tau A_1\} b \ddot{u}_{id}(\tau) d\tau \\ &= -A_1^{-2} \exp\{tA_1\} [a_0 - \dot{x}_d(0)] \\ &\quad + \int_0^t \exp\{-\tau A_1\} \frac{d^2}{d\tau^2} (A_1 x_d + b u_{id} - \dot{x}_d) d\tau \\ &= -A_1^{-2} \exp\{tA_1\} [a_0 - \dot{x}_d(0)] \end{aligned}$$

The only source of error is due to initial conditions  $x_0, a_0$ , which may be canceled exactly by appropriate choice of  $u_0, \dot{u}_0$  since  $b, A_1 b$  are linearly independent (the system is *controllable*). If not canceled exactly, the term must decay asymptotically under the assumption that  $k_1, k_2$  are stabilizing gains. Tracking is perfect, or at least asymptotically perfect, for any arbitrary input.

This strategy resembles the first open-loop control scheme introduced above in that it assumes perfect information regarding the plant dynamics. Any uncertainty in the model, its parameters, or the presence of noise will invalidate the result. Moreover, since it requires derivatives of the reference trajectory,  $\dot{x}_d$ , the scheme would be practicable only in cases where the entire reference trajectory is known in advance: differentiating unknown signals on-line generally results in unacceptably large noise amplitudes.

**Robust Tracking.** It was seen above that the effect on the steady-state response of bounded noise perturbations could be made arbitrarily small through the use of high-gain feedback. Subsequently, it was shown that high-gain feedback increases the rate at which the system tends toward its steady state. Here, these insights will be combined and an attempt made to track  $x_d(t)$  as if it were a moving set point through the continued exploitation of high-gain feedback. Intuitively, it is hoped that the resulting tendency to steady state will be "faster" than the rate of change of the set point. As usual, the advantage attending the reliance on intrinsic stability properties of the system will be the reduced need for a priori information.

Let the feedback control be chosen using the "high-gain" philosophy as in Eq. 2. If the precompensating function is simply set to be proportional to the desired position,

$$\Gamma_p[x_d] = \gamma_1 \theta_d$$

according to Eq. 10,

$$e(t) = \begin{bmatrix} \dot{\theta}_d \gamma_2 \\ \gamma_1 \\ 0 \end{bmatrix} - A_1^{-2} \exp\{tA_1\} [a_0 + \int_0^t \exp\{-\tau A_1\} \ddot{\theta}_d(\tau) d\tau]$$

and, not surprisingly, there are terms that contribute error in proportion to the desired velocity and "time-averaged" desired acceleration. Note that if the feedback gains are increased

according to the high-gain policy, namely,  $\gamma_1 \approx \gamma_2^2$ , it seems as though high-gain feedback may reduce the error introduced by velocity. This intuition may be rigorously confirmed, again by appeal to a Lyapunov argument after making the further assumption that there is some (in general unknown) bound on the desired speed,

$$|\dot{\theta}_d| \leq \rho$$

Defining the modified error coordinates,

$$\tilde{e} \triangleq \begin{bmatrix} \theta_d - \theta \\ \dot{\theta} \end{bmatrix}$$

the closed-loop system takes the form

$$\dot{\tilde{e}} = A_1 \tilde{e} - d$$

where  $d \triangleq [\dot{\theta}_d]$ . Consider, then, the Lyapunov candidate,

$$v(\tilde{e}) \triangleq \frac{1}{2} \tilde{e}^T P \tilde{e}$$

where

$$P \triangleq \begin{bmatrix} \gamma_1 & \frac{1}{2} \gamma_2 \\ \frac{1}{2} \gamma_2 & M \end{bmatrix}$$

(the same matrix defined above) is positive definite as long as the relation 5 is satisfied. The derivative along the solutions of the closed-loop error equation is

$$\begin{aligned} \dot{v} &= \tilde{e}^T P \dot{\tilde{e}} = \tilde{e}^T P \left[ -\frac{\gamma_2}{2M} \tilde{e} - d \right] \\ &\leq -\frac{\gamma_2}{4M} \tilde{e}^T P \tilde{e} + \frac{M}{\gamma_2} d^T P^{-1} d \\ &= -\frac{\gamma_2}{4M} \left[ \tilde{e}^T P \tilde{e} + \frac{4M^2 \dot{\theta}_d^2}{\gamma_2^2 (M\gamma_1 - \gamma_2^2/4)} \right] \end{aligned}$$

If the reference signal is assumed to have a bounded derivative,  $\dot{\theta}_d \leq \rho$ , for some constant scalar  $\rho$ , the derivative is negative whenever

$$\|e\|^2 \geq \frac{4M^2 \rho^2}{\|P\| \gamma_2^2 (4M\gamma_1 - \gamma_2^2)}$$

and it follows that the error magnitude decreases monotonically at least until it reaches the constant term in this inequality. The latter may be made arbitrarily small by increasing the magnitude of the feedback gains,  $\gamma_1, \gamma_2$ .

**Adaptive Control.** The final approach to the linear-time-invariant servo problem to be considered in this entry might be said to marshall the power of schemes such as pole placement or inverse dynamics without requiring exact a priori knowledge of the system parameters. The field of adaptive control is relatively new since the first convergence results for general linear-time-invariant systems were reported only in 1980 (12). The method described here falls within the class of "model reference" schemes (13,14).

Suppose a desired trajectory,  $x_d$ , has been specified along with a precompensating feedforward law,  $u_{pc} \triangleq \Gamma[x_d]$ , which forces a known model

$$\dot{x}_m = A_m x_m + b_m u_{pc}$$

to track  $x_d$  acceptably. Suppose, moreover, that the true system to be controlled,

$$\dot{x} = Ax + bu$$

satisfies the condition  $\gamma b = b_m$  for some (unknown) positive scalar,  $\gamma > 0$ , and is known to admit a feedback law,  $u_{fb} \triangleq [\alpha, \beta]x$  such that the closed-loop system yields the model behavior,  $A + b[\alpha, \beta] = A_m$ , for some (unknown) set of gains,  $\alpha, \beta \in \mathbf{R}$ . The adaptive control law takes the form

$$u_{ad} \triangleq \hat{k}^T(t) \begin{bmatrix} x \\ u_{pc} \end{bmatrix}$$

where  $\hat{k}(t)^T = [\hat{\alpha}(t), \hat{\beta}(t), \hat{\gamma}(t)]$  denotes a set of gain estimates that will be continuously adjusted on the basis of observed performance. Let  $k^T = [\alpha, \beta, \gamma]$  denote an unknown "true" vector of gains. The closed-loop system resulting from  $u_{ad}$  may be written in the form

$$\dot{x} = A_m x + b_m u_{pc} + [\hat{k}(t) - k]^T \begin{bmatrix} x \\ u_{pc} \end{bmatrix}$$

hence, defining the state error coordinates,  $e \triangleq x_m - x$ , and the "parameter error" coordinates,  $\tilde{k} \triangleq k - \hat{k}$ , yields the system

$$\dot{e} = A_m e - \frac{1}{\gamma} b_m \tilde{k}^T \begin{bmatrix} x \\ u_{pc} \end{bmatrix}$$

Notice, further, that  $\dot{\tilde{k}} = -\dot{\hat{k}}$ , hence, adjustments in  $\hat{k}$  afford exact adjustments of the opposite sign in the parameter error vector. The question remains, then, concerning the choice of an "adaptive law,"  $\dot{\hat{k}} = f(\hat{k}, e, u_{pc})$  that will make the complete error system converge.

At the very least, we may assume that  $A_m$  defines an asymptotically stable closed-loop system since the model is capable of tracking  $x_d$  in the first place. According to the theory of Lyapunov (15), it is therefore guaranteed that a positive definite symmetric matrix,  $P_m$ , exists such that  $x^T P_m A_m x \leq 0$  with equality only for  $x = 0$ . Find such a  $P_m$ , and set the adaptive law as

$$\dot{\hat{k}} = - \begin{bmatrix} x \\ u_{pc} \end{bmatrix} e^T P_m b_m$$

To show that the resulting closed-loop error equations converge, consider the extended Lyapunov candidate  $v(e, \tilde{k}) \triangleq \frac{1}{2} [e^T P_m e + 1/\gamma \tilde{k}^T \tilde{k}]$ . Since

$$\begin{aligned} \dot{v} &= e^T P_m A_m e - \frac{1}{\gamma} e^T P_m b_m \tilde{k}^T \begin{bmatrix} x \\ u_{pc} \end{bmatrix} + \frac{1}{\gamma} \tilde{k}^T \begin{bmatrix} x \\ u_{pc} \end{bmatrix} e^T P_m b_m \\ &= e^T P_m A_m e \leq 0 \end{aligned}$$

it is guaranteed that  $v$ , and therefore  $e, \tilde{k}$ , remain bounded for all time. A further technical argument based on the assumption that  $u$  is bounded for all time may be used to finally show that  $e$  actually converges to zero as well (16).

Now apply this general method to the particular system representing the one-degree-of-freedom robot (Eq. 1) using the model resulting from the pole-placement feedback scheme,

$$A_m \triangleq \begin{bmatrix} 0 & 1 \\ -k_1 & -k_2 \end{bmatrix} \quad b_m \triangleq \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

In this particular context the required assumptions outlined above are easily seen to hold. The "true" gain settings that make the closed-loop system behave like the model are given as  $\alpha = -Mk_1, \beta = -Mk_2, \gamma = M$ , and the assumption that  $M > 0$  is unexceptionable. Note that

$$P_m \triangleq \begin{bmatrix} k_1 & \frac{1}{2} k_2 \\ \frac{1}{2} k_2 & 1 \end{bmatrix}$$



satisfies the conditions listed above for the Lyapunov matrix—it is positive definite (as long as  $4k_1 > k_2^2$ ); and its product with  $A_m$  defines a negative definite symmetric matrix. Thus, the full adaptive controller takes the form

$$u_{ad} \triangleq \hat{\alpha}(t)\theta + \hat{\beta}(t)\dot{\theta} + \hat{\gamma}(t)u_{pc}$$

where

$$\frac{d}{dt} \begin{bmatrix} \hat{\alpha}(t) \\ \hat{\beta}(t) \\ \hat{\gamma}(t) \end{bmatrix} = [\frac{1}{2}(\theta - \theta_m)k_2 + (\dot{\theta} - \dot{\theta}_m)]$$

### General Robot-Arm Dynamics

This section generalizes the discussion to the case of multi-jointed open kinematic chains. First, a brief but fairly general treatment of kinematics affords quick derivation of the general rigid-body model of robot-arm dynamics. In reality, this model is a simplification of empirically observed phenomena. Depending on what class of robot manipulator one studies, additional nonlinearities and dynamics cannot be ignored, and several examples of such phenomena are examined briefly.

**Rigid-Body Model: Lagrangian Formulation of Newton's Laws.** Contemporary robots are built to be rigid, and models of their idealized behavior are based on the geometry of rigid transformations. After reviewing some elementary facts leading to a useful algebraic formalism for manipulating objects that obey this “extrinsic” geometry, robot kinematics—the “intrinsic” geometry of robots—will be investigated and both geometric domains will be used to understand the dynamical properties of robot motion.

**Rigid Transformations and Frames of Reference.** For purposes of this entry, the physical world is an *affine space*,  $\mathbf{A}^3$ : each element is a point described by three real numbers; however, there is no predefined origin (17). By taking differences between elements of the affine space,  $a, b \in \mathbf{A}^3$ , we obtain elements of *Euclidean vector space*,  $\overrightarrow{ab} \in \mathbf{E}^3$ , with vector addition, scalar multiplication, inner product, and norm (17,18). Define a *rigid transformation* to be a continuous transformation of affine 3-space which preserves distance:

$$\mathcal{T} \triangleq \{\tau \in C^0[\mathbf{A}^3, \mathbf{A}^3]: \|\overrightarrow{\tau(a)\tau(b)}\| = \|\overrightarrow{ab}\| \text{ for all } a, b \in \mathbf{A}^3\}$$

Examples of rigid transformations include the vector translations,

$$\mathcal{V} \triangleq \{\tau_v \in \mathcal{T}: \text{for some } v \in \mathbf{E}^3, \tau_v(a) = b \Rightarrow \overrightarrow{ab} = v \text{ for all } a, b \in \mathbf{A}^3\}$$

and rotations around fixed points

$$\mathcal{R} \triangleq \{\tau_{(R,o)} \in \mathcal{T}: \text{for some } R \in \mathcal{SO}(3), o \in \mathbf{A}^3, \tau_{(R,o)}(o) = o, \text{ and } \overrightarrow{o\tau_{(R,o)}(b)} = R\overrightarrow{ab} \text{ for all } b \in \mathbf{A}^3\}$$

where  $\mathcal{SO}(3)$  is the set of orthogonal linear operators on  $\mathbf{E}^3$  with positive determinant. In fact, it can be shown (18,19) that these examples constitute the entirety of  $\mathcal{R}$ : that is, for any arbitrarily chosen “origin,”  $o \in \mathbf{A}^3$ , every rigid transformation of  $\mathbf{A}^3$  may be uniquely expressed as the composition of a translation with a rotation around  $o$ :  $\tau = \tau_v \circ \tau_{(R,o)}$ . Thus, once a fixed point or “origin,”  $o$ , has been chosen, the set of rigid transformations may be put into one-to-one correspondence with the set of translations and rotations:  $\mathcal{T} \approx \mathcal{W} \triangleq \mathbf{R}^3 \times$

$\mathcal{SO}(3)$ . Since  $\tau_v^{-1} = \tau_{-v}$  and  $\tau_R^{-1} = \tau_{R^{-1}} = \tau_{R^T}$ , it follows that  $\tau^{-1} = \tau_{R^T} \circ \tau_{-v}$  is always defined, and hence  $\mathcal{T}$  is a group.

The position and orientation of any rigid body may be precisely described by fixing four points of  $\mathbf{A}^3$  called a *frame of reference*

$$\mathcal{F} \triangleq \{a, b, c, o\}$$

where

$$\mathcal{B} = \{x, y, z\} \triangleq \{\overrightarrow{oa}, \overrightarrow{ob}, \overrightarrow{oc}\}$$

is a “right-handed” orthonormal basis of  $\mathbf{E}^3$ . That is, the direction of  $z$  on the line orthogonal to the  $xy$  plane of  $\mathbf{E}^3$  obeys a right-hand rule with respect to rotations on that plane. Note that the image of a frame under a rigid transformation,  $\tau(\mathcal{F}_1) = \{\tau(a_1), \tau(b_1), \tau(c_1), \tau(o_1)\}$  is also a frame since, e.g.,  $\langle \tau(o)\tau(a) | \tau(o)\tau(b) \rangle = \langle \overrightarrow{oa} | \overrightarrow{ob} \rangle = 0$  in consequence of the parallelogram rule relating norms and inner products. On the other hand, if  $\mathcal{F}_1, \mathcal{F}_2$  are two frames, there exists an orthogonal transformation with positive determinant,  $R \in \mathcal{SO}(3)$  such that  $x_2 = Rx_1, y_2 = Ry_1, z_2 = Rz_1$  since both sets of vectors define a right-hand orthonormal basis. Thus, defining  $\tau_{12} \triangleq \tau_{o_1o_2} \circ \tau_{(R,o_1)}$ , there is  $\tau_{12}(o_1) = \tau_{o_1o_2}(o_1) = o_2$ , and, e.g.,

$$\begin{aligned} \overrightarrow{a_2\tau_{12}(a_1)} &= \overrightarrow{o_2\tau_{12}(a_1)} - \overrightarrow{o_2a_2} \\ &= \tau_{12}(o_1)\tau_{12}(a_1) - x_2 \\ &= o_1\tau_{R,o_1}(a_1) - x_2 \\ &= 0 \end{aligned}$$

so that  $\mathcal{F}_2 = \tau_{12}(\mathcal{F}_1)$ . Thus, an exact description of the change in position and orientation of a given rigid body is provided equivalently by fixing a frame in the body and specifying either a rigid transformation or a second frame. Alternatively, given any two rigid bodies, an exact description of their relative position and orientation is provided by fixing a frame in each body. Assuming the existence of some “base frame,”  $\mathcal{F}_0$ , we now have a model of any other rigid position and orientation in terms of  $\mathcal{W} \triangleq \mathbf{R}^3 \times \mathcal{SO}(3)$ , which we will call “work space.”

It has become a tradition in robotics to use *homogeneous coordinates* in the representation of objects in  $\mathbf{A}^3$ . Since rigid transformations (which, of course, are affine rather than linear) admit a matrix representation in these coordinates, the resulting simplicity in notation seems worth the slight conceptual complication. Homogeneous coordinates result in a more complicated representation from the numerical point of view as well. Thus, the *matrix representation of a point*,  $p \in \mathbf{A}^3$ , with respect to the any frame,  $\mathcal{F}_j = \{o_j, a_j, b_j, c_j\}$ , is denoted

$${}^j p = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \end{bmatrix}$$

and is understood to represent the geometric relation

$$\pi_4 \overrightarrow{o_j p} = \pi_1 x_j + \pi_2 y_j + \pi_3 z_j$$

Note that two arrays in  $\mathbf{R}^4$  represent the same point with respect to the same frame if they are scalar multiples. The attending conceptual complications arise because it is useful, at the same time, to employ matrix representations of vectors in  $\mathbf{E}^3$ . Confusion between the previous arrays and these may be avoided by treating the latter as constituting “ideal points”

of  $\mathbf{A}^3$  which, together, comprise projective space,  $\mathbf{P}^3$  (18). Thus, if  $v \in \mathbf{E}^3$ ,

$${}^j v = \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ 0 \end{bmatrix}$$

represents the geometric relation

$$v = \nu_1 x_j + \nu_2 y_j + \nu_3 z_j$$

It follows that for any  $q, p \in \mathbf{A}^3$ , if  ${}^j q = [\xi_1, \xi_2, \xi_3, \xi_4]^T, {}^j p = [\pi_1, \pi_2, \pi_3, \pi_4]^T$ ,

$${}^j \overrightarrow{pq} = \frac{1}{\xi_4} {}^j q - \frac{1}{\pi_4} {}^j p$$

Finally, define the *matrix representation of a frame*,  $\mathcal{F}_i$  with respect to  $\mathcal{F}_j$  to be the  $4 \times 4$  array whose columns are

$${}^j F_i \triangleq [{}^j x_i, {}^j y_i, {}^j z_i, {}^j o_i \frac{1}{\omega_4}]$$

where  $\omega_4$  is the fourth component of  ${}^j o_i$ . According to the discussion of the previous paragraph, this definition is interchangeable with the following: the *matrix representation of a rigid transformation*,  $\tau \in \mathcal{W}$  with respect to frame  $\mathcal{F}_j$  is given by  ${}^j \mathcal{F}_i$ , where  $\mathcal{F}_i = \tau(\mathcal{F}_j)$ .

**Kinematics.** Although we are most interested in the actions of a robot in work space,  $\mathcal{W}$ , commands are effected through a set of joints that constrain the relative motion of a sequence of rigid links—a *kinematic chain*. Kinematics, the study of spatial relationships between such a configuration of mechanically interconnected and constrained rigid bodies, is a very old discipline, and it is not possible to provide more than a superficial account of its role in robot control. A much more thorough treatment of these considerations is provided in the general robotics literature (20,21).

The position of a kinematic chain possessed of  $n - k$  prismatic and  $k$  revolute joints may be described as a point,  $q$ , in *joint space*, the cross product of  $\mathbf{R}^{n-k}$  with a  $k$ -dimensional torus. There are two circumstances under which  $\mathcal{J}$  may be accurately modeled as a subset of  $\mathbf{R}^n$ . If full revolution is mechanically possible but joint sensors are available that transduce angular displacement with respect to some absolute position, e.g., from which a revolution and a half displacement is read as  $540^\circ$  rather than  $180^\circ$ , then this is clearly the case. If each revolute joint is mechanically constrained to prevent a full  $360^\circ$  revolution, then any reasonable set of position sensors return a bounded linear measurement of joint displacement. We will assume that one of these situations prevails, thereby assuring that  $\mathcal{J} \subset \mathbf{R}^n$ .

A *joint transformation* is a map from  $\mathbf{R}$  into  $\mathcal{W}$  that relates a coordinate system fixed in link  $i - 1$  to one fixed in link  $i$  through the action of the  $i$ th joint. According to the standard conventions, the joint transformation depends on three parameters that describe the link and one variable—the  $i$ th coordinate of  $\mathcal{J}$ . Since the link is rigid, these frames are related by a rigid transformation whose matrix representation may be written in the form

$${}^{i-1} \mathcal{F}_i = \begin{bmatrix} \cos \theta_i & -\cos \alpha_i \sin \theta_i & -\sin \alpha_i \sin \theta_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \alpha_i \cos \theta_i & \sin \alpha_i \cos \theta_i & a_i \sin \theta_i \\ 0 & \sin \alpha_i & \cos \alpha_i & \delta_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where either  $\theta_i$  or  $\delta_i$  is the joint variable depending on whether the joint is revolute or prismatic, respectively, and the other kinematic parameters are defined in the link body, e.g., as in Ref. 20.

More generally, a *kinematic transformation* is a map

$$g: \mathcal{J} \rightarrow \mathcal{W}$$

which is the group product of  $n$  joint transformations, representing the rigid transformation required to align the “base frame” with the “end-effector frame”:

$$g(q) = {}^0 F_1(q_1) {}^1 F_2(q_2) \cdots {}^{n-1} F_n(q_n)$$

**Dynamics.** The rigid-body model of robot-arm dynamics may be most quickly derived by appeal to the Lagrangian formulation of Newton's equations. If a scalar function, termed a *Lagrangian*,  $\lambda = \kappa - \nu$ , is defined as the difference between total kinetic energy  $\kappa$  and total potential energy  $\nu$  in a system, the equations of motion obtain from

$$\frac{d}{dt} D_q \lambda - D_q \lambda = \tau^T$$

where  $\tau$  is a vector of external torques and forces (17,22).

First consider the kinetic energy contributed by a small volume of mass  $\delta m_i$  at position  $p$  in link  $\mathcal{L}_i$ ,

$$\delta \kappa_i = \frac{1}{2} {}^0 \dot{p}_i^T {}^0 \dot{p}_i \delta m_i$$

where  ${}^0 p_i = {}^0 F_i {}^i p$  is the matrix representation of the position  $p$  in the base frame of reference,  ${}^0 F_i$  is the matrix representation of the frame of reference of link  $\mathcal{L}_i$  in the base frame, and  ${}^i p$  is the matrix representation of the point in the link frame of reference, and, hence,

$$\dot{p}_i = \dot{F}_i {}^i p$$

since the position in the body is independent of the generalized coordinates. (We will omit the prior superscript 0 when it is clear that the coordinate system of reference is the base.) The total kinetic energy contributed by this link may now be written

$$\begin{aligned} \kappa_i &= \int_{\mathcal{L}_i} \frac{1}{2} [\dot{F}_i {}^i p]^T \dot{F}_i {}^i p \, dm_i \\ &= \int_{\mathcal{L}_i} \frac{1}{2} \text{tr} \{ \dot{F}_i {}^i p [\dot{F}_i {}^i p]^T \} \, dm_i \\ &= \frac{1}{2} \text{tr} \{ \dot{F}_i \int_{\mathcal{L}_i} p {}^i p^T \, dm_i [\dot{F}_i]^T \} \\ &= \frac{1}{2} \text{tr} \{ \dot{F}_i \bar{P}_i \dot{F}_i^T \} \end{aligned}$$

(since the frame matrix is constant over the integration), where  $\bar{P}_i$  is a symmetric matrix of *dynamic parameters* for the link. Explicitly, if the link has mass  $\bar{\mu}_i$ , center of gravity (in the local link coordinate system)  $\bar{p}_i$ , and inertia matrix  $\bar{J}_i$ ,

$$\bar{P}_i \triangleq \begin{bmatrix} \bar{J}_i & \bar{\mu}_i \bar{p}_i \\ \bar{\mu}_i \bar{p}_i^T & \bar{\mu}_i \end{bmatrix}$$

Passing to the *stack representation* (refer to Appendix A)

$$\begin{aligned} 2\kappa_i &= \text{tr} \{ \dot{F}_i \bar{P}_i \dot{F}_i^T \} \\ &= [(\dot{F}_i \bar{P}_i)^S]^T \dot{F}_i^S \\ &= [(\bar{P}_i \otimes I)^T \dot{F}_i^S] \\ &= [\bar{P}_i^S]^T \dot{\bar{P}}_i \dot{F}_i^S \\ &= [(D_q F_i^S) \dot{q}]^T \dot{\bar{P}}_i (D_q F_i^S) \dot{q} \\ &= \dot{q}^T M_i \dot{q} \end{aligned}$$

where we have implicitly defined

$$M_i(q) \triangleq [D_q F_i^S]^T \bar{P}_i D_q F_i^S \quad \bar{P}_i \triangleq \bar{P}_i^T \otimes I$$

It follows that the total kinetic energy of the entire chain is given as

$$\kappa = \frac{1}{2} \dot{q}^T M(q) \dot{q} \quad M(q) \triangleq \sum_{i=1}^n M_i(q)$$

The potential energy contributed by  $\delta m_i$  in  $\mathcal{L}_i$  is

$$\delta v_i = z_0^T F_i \int_{\lambda_i} p g \, \delta m_i$$

where  $g$  is the acceleration of gravity; hence, the potential energy contributed by the entire link is

$$v_i = z_0^T F_i \int_{\lambda_i} p g \, dm_i = z_0^T F_i \bar{p}_i g$$

and  $v = \sum_{i=1}^n z_0^T F_i \bar{p}_i g$ . Assume that  $z_0$  "points up" in a direction opposing the gravitational field.

To proceed with the computation, note that  $D_{\dot{q}} \lambda = D_{\dot{q}} \kappa = \dot{q}^T M(q)$ ; hence,

$$\frac{d}{dt} D_{\dot{q}} \lambda = \dot{q}^T M(q) + \dot{q}^T \dot{M}(q)$$

Moreover,

$$\begin{aligned} D_{\dot{q}} \kappa &= \frac{1}{2} \dot{q}^T D_{\dot{q}} [M(q) \dot{q}] \\ &= \frac{1}{2} \dot{q}^T [\dot{q}^T \otimes I] D_{\dot{q}} M^S \end{aligned}$$

hence, if all terms from Lagrange's equation involving the generalized velocity are collected, we may express them in the form  $\dot{q}^T B^T$ , where

$$B(q, \dot{q})^T \triangleq \dot{M}(q) - \frac{1}{2} [\dot{q}^T \otimes I] D_{\dot{q}} M^S$$

Finally, by defining  $k(q) \triangleq [D_{\dot{q}} v]^T$ , Lagrange's equation may be written in the form

$$M(q) \ddot{q} + B(q, \dot{q}) \dot{q} + k(q) = \tau \quad (11)$$

The "inertia" matrix  $M$  may be shown to be positive definite over the entire work space as well as bounded from above since it contains only polynomials involving transcendental functions of  $q$ , and  $B$  contains terms arising from "coriolis" and "centripetal" forces and hence is linear in  $\dot{q}$  (these forces are quadratic in the generalized velocities), and bounded in  $q$ , since it involves only polynomials of transcendental functions in the generalized position. Finally,  $k$  arises from gravitational forces, is bounded, and may be observed to have much simpler structure (still polynomial in transcendental terms involving  $q$ ) than the other expressions. An important study of the form of these terms was conducted by Bejczy (23).

To bring this into the state-space form discussed above, let  $x \triangleq (q, \dot{q})^T$ ,  $u \triangleq \tau$ , to get

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -M^{-1}(x_1) [B(x_1, x_2) x_2 + k(x_1) - u] \end{aligned} \quad (12)$$

with output map provided by the kinematics,

$$y = g(x)$$

**Omissions in Rigid-Body Model.** The theoretical presentation below will rest almost entirely for rigorous justification on the rigid-body dynamic model derived above. In fact, most commercially available robots deviate from this model quite

dramatically. Flexibility in the links, slippage in transmissions, and backlash in gear trains introduce stiction, hysteresis, and other nonlinear effects. Harmonic drives and compliance at the joints introduce extra dynamics (24). Demagnetization and potential damage to the windings place limits on the maximum permissible armature current and, therefore, output torque, of a dc servo. Moreover, although it is traditional in the control community to model electric motors as if they were first-order lags (25), it is not impossible to find commercial robot arms employing dc servos whose mechanical and electrical time constants of similar magnitude, and which, in consequence, have second-order dynamics, not uncommonly *oscillatory* (26). Thus, not only may the model introduced in Eq. 12 have missing functional terms in practice but also its dimensionality,  $2n$ , may too low by at least again as much as the number of actuators,  $n$ .

Unfortunately, it does not seem likely that a better general model will be available in the near future. There is no generally accepted understanding of which dynamic effects are significant and which may be ignored beyond the rigid-body model (Eq. 12) [which, itself, is not universally acknowledged to be of greatest importance (24)]. In part, this is due to the great diversity of kinematic, actuator, and sensory arrangements that may be found on the commercial market. In part, it is a reflection of the relative novelty of the field. Similarly, there is insufficient understanding of the disturbances resulting from digital implementation of control algorithms—quantization and roundoff errors—to admit of any reliable model for these effects. Even in the control community itself researchers are only beginning to come to terms with quantization problems (27).

In this section models for some of the important dynamic and nonlinear disturbances not captured in Ref. 12 will be introduced. The relative importance of any of these discrepancies can only be determined by empirical investigation in the context of a specific mechanical apparatus. It is worth remarking here that the utility of much of the theoretical work to be presented in subsequent sections will require empirical verification for the same reasons. Clearly, in the absence of trustworthy models, proofs (and even simulations) alone are not terribly convincing.

**Additional Dynamics.** In the rigid-body model developed above the control input appears as a set of torques,  $\tau$ , injected at each joint independently. In reality, all robotic actuators that deliver torque or force to a joint are themselves commanded by a reference voltage computed by the controller. For a significant class of actuators, the torque or force output is not a simple function of this reference voltage but may itself involve a dynamic relationship. This may be seen most easily through a specific example.

A typical actuator for a revolute joint is the dc servo. A dc motor converts electrical to mechanical power through the exchange of energy in two sets of *windings* via electromagnetic forces (25). For a typical motor the torque delivered to the output shaft is roughly proportional to the current in the "armature windings,"  $\tau_g = K_a I_a$ , and this is exactly balanced by the d'Alembert torque due the angular acceleration of motor inertia,  $\tau_m = J \ddot{\theta}$ , as well as the external load torque,  $\tau_l$ , placed on the motor (25),

$$\tau_m + \tau_l = \tau_g$$

On the other hand, the current in the armature winding results from the application of some command voltage,  $v_a$ ,

through the armature resistance and inductance,  $R_a$ ,  $L_a$ , opposed by the back-generated voltage  $v_b = K_b \dot{\theta}$ ,

$$L_a \frac{di_a}{dt} + R_a i_a + v_b = v_a$$

There results the second-order linear differential equation (25)

$$\begin{aligned} \frac{d\dot{\theta}}{dt} &= \kappa_m \dot{\theta} + \gamma_m i_a + \xi_m \tau_l \\ \frac{di_a}{dt} &= \kappa_e i_a + \gamma_e \dot{\theta} + \xi_e v_a \end{aligned}$$

If the "electric time constant,"  $\kappa_e$ , is a very large negative number in comparison to the "mechanical time constant,"  $\kappa_m$ , the second equation is really algebraic,

$$R_a i_a + v_b = v_a$$

and the relationship between command voltage and generated torque is given by

$$\begin{aligned} \mu \frac{d\dot{\theta}}{dt} &= \beta \dot{\theta} + \phi v_a - \tau_l \\ \mu &\triangleq J \quad \beta \triangleq \frac{K_\tau K_b}{R_a} \quad \phi \triangleq \frac{K_\tau}{R_a} \end{aligned}$$

This is typically the case for common dc servo motors (25), although important exceptions have been noted in the literature (26).

First suppose direct-drive arm, and  $n$  such actuators are mounted directly at each joint being controlled. It follows that  $\tau_l = \tau + j$ , the load torque seen by the  $j$ th actuator, is exactly the  $j$ th component of the rigid-body external torque vector,  $\tau$ , of Eq. 11. Let

$$M_m \triangleq \begin{bmatrix} \mu_1 & 0 \\ & \ddots \\ 0 & \mu_n \end{bmatrix}; V_m \triangleq \begin{bmatrix} \phi_1 & 0 \\ & \ddots \\ 0 & \phi_n \end{bmatrix}; B_m \triangleq \begin{bmatrix} \beta_1 & 0 \\ & \ddots \\ 0 & \beta_n \end{bmatrix}$$

be the diagonal coefficient matrices corresponding to the  $n$  decoupled motors, and let  $v \triangleq [v_{a1}, \dots, v_{an}]^T$  be the vector of command voltages at each joint. Then according to the reasoning of Dynamics,

$$[M(q) + M_m] \ddot{q} + (B(q, \dot{q}) + B_m) \dot{q} + k(q) = V_m v \quad (13)$$

producing a state-space system with very similar characteristics to the original Eq. 12. It is worth noting that most of the theoretical results discussed below are still valid in this case.

Suppose, on the other hand, that an actuator is mechanically coupled to each joint  $j$  via some mechanical transmission—a chain, a harmonic drive, a gear train—as is the case with most commercially available robots. In this case,  $\tau_l$ , the load torque seen by the motor, is that delivered from the joint along the transmission. As a first-order approximation, the transmission will be modeled as a passive intermediate inertial load,  $\mu_t$ , lumped at the joint end of the transmission on a shaft with torsional spring constant  $\kappa_t$  and torsional damping constant  $\beta_t$ . Letting  $\rho_j$  denote the position of the rotor, and  $\theta_j$  the position of the joint, this assumption may be written as

$$\tau_l \triangleq \kappa_t (\rho_j - \theta_j) + \beta_t (\dot{\rho}_j - \dot{\theta}_j)$$

with  $\tau_t \triangleq \mu_t \ddot{\theta}_j$  comprising the d'Alembert torque due to acceleration of the transmission inertia. The latter is balanced by the transmitted motor torque,  $\tau_l$ , and the nonlinear coupling torques,  $\tau_j$ ; thus, the complete torque equations are

$$\tau_g = \tau_m + \tau_l \quad \tau_l = \tau_t + \tau_j$$

Now define the transmission diagonal arrays,  $M_t$ ,  $B_t$  in terms of the inertial and viscous damping coefficients, respectively, as before. Define the motor shaft angle vector  $r \triangleq [\rho_1, \dots, \rho_n]^T$ . The torque balance equations take the vector form

$$-B_m \dot{r} + V_m v = M_m \ddot{r} + B_t (\dot{r} - \dot{q}) + K_t (r - q)$$

$$B_t (\dot{r} - \dot{q}) + K_t (r - q) = (M(q) + M_t) \ddot{q} + B(q, \dot{q}) \dot{q} + k(q)$$

Writing this in state-space notation yields a much more complex dynamic system. Letting

$$z = \begin{bmatrix} q \\ r \\ \dot{q} \\ \dot{r} \end{bmatrix} \triangleq \begin{bmatrix} z_{11} \\ z_{12} \\ z_{21} \\ z_{22} \end{bmatrix}$$

yields

$$z_{11} = z_{21}$$

$$z_{12} = z_{22}$$

$$z_{21} = -[M + M_t]^{-1} [(B + B_t) z_{21} + K_t z_{11} - B_t z_{22} - K_t z_{12} + k(z_{11})]$$

$$z_{22} = -M_m^{-1} (B_m + B_t) z_{22} + K_t q_{12} - B_t z_{21} - K_t z_{11} + V_m v]$$

**Local Nonlinearities.** As well as (at least) doubling the dimensionality of the underlying dynamics, the presence of a mechanical transmission introduces a variety of memoryless nonlinearities—backlash, hysteresis, etc.—that depend very much on the nature of the mechanism. Returning to the direct-drive arm, probably the two most significant sources of nonlinearities distinct from those due the rigid-body equations (Eq. 11) are friction and saturation.

A solid object in contact with any surface is subject to a variety of frictional forces that may be observed, in general, to vary with its velocity relative to that surface. The simplest of these is *viscous damping*, a force exerted in the opposite direction of motion in direct proportion,  $\beta_v$ , to the velocity magnitude. Further, it may be observed that at low speeds the magnitude of these opposing forces ceases to diminish beyond a certain level, and hence, a constant term,  $\beta_c$ , called *Coulomb friction* must be added to the viscous term. Finally, the force  $\beta_s$  required to bring a motionless object to some nonzero velocity typically exceeds that needed to overcome the friction forces at nonzero velocity: this is termed *stiction*. Thus, an appropriate model for actually observed frictional forces might be given as

$$\tau_{\text{frict}}(q, \dot{q}) = \beta_c + \beta_v \dot{q} + \beta_s \delta_0(\dot{q})$$

Recall that  $\delta_0$  denotes the delta function introduced earlier.

The second source of additional nonlinearities unmodeled in the previous section is a consequence of the fact that all real devices can deliver only finite power. In this light the admissible set of controls must be modified to include magnitude con-

straints, most easily modeled in the form of a saturation non-linearity on the command input at the  $j$ th joint,

$$s_i(v_j) \triangleq \begin{cases} \bar{s}_i & v_j \geq \bar{s}_i \\ v_j & v_j \in (\underline{s}_i, \bar{s}_i) \\ \underline{s}_i & v_j \leq \underline{s}_i \end{cases}$$

These forces act on each joint independent of the motion of the others (except, of course, through dynamic coupling of velocities) and are termed "local disturbances" for that reason. Let  $B_c$ ,  $B_v$ ,  $B_s$  be diagonal matrices containing, respectively, the Coulomb, viscous, and stiction coefficients for each joint, let  $d(\dot{q}) = [\delta_0(\dot{q}_1), \dots, \delta_0(\dot{q}_n)]^T$  be the vector of delta functions in each generalized velocity, and let  $s(v) \triangleq [s_1(v_1), \dots, s_n(v_n)]^T$  be the vector of saturation nonlinearities. Combining these within the vector of external disturbances,  $\tau$  in Eq. 13, yields a new version of the state-space dynamics of the form

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -[M(x_1) + M_m(x_1)]^{-1}[(B(x_1, x_2) + B_m + B_v)x_2 \\ &\quad + B_s d(\dot{q}) + k(x_1) + B_c - V_m s(u)] \end{aligned} \quad (14)$$

### Feedback Control of General Robot Arms

**Feedback Control: Behavior of Error-Driven Systems** presented an account of the behavior of classical feedback controllers in the context of set-point regulation. Here, the attempt is made to generalize that account in two rather different directions. First, the breadth of task domain is considerably widened beyond set-point regulation, and the possibility is explored of formalizing a task-encoding methodology based on feedback control structures that generalize the error-driven characteristics of the PD controller. Second, the underlying dynamics of the system to be controlled are specified by the  $n$ -degree-of-freedom rigid-body model (Eq. 12) of which the simple pendulum (Eq. 1) was a particularly easy example.

Three successively more generalized methods of task encoding are presented beyond the set-point error introduced above: specification in terms of the extrema of objective functions; in terms of the "fall lines" of gradient vector fields arising from objective functions; in terms of general first-order dynamics. By interpreting an objective function as potential energy, its gradient is shown to determine a stabilizing feedback control structure for a general robot arm. The implications of this result for achieving the tasks specified by the various encoding methods are discussed.

**A Generalized Robot Task-Encoding Methodology.** By a "task-encoding methodology" is meant any procedure through which an abstract goal is translated into robot-control strategies resulting in its achievement. Here, we explore techniques that do not involve task specification via reference trajectory. In part, the exploration is motivated by the difficulties involved in generalizing classical servo theory to the rigid-body dynamics (Eq. 12), as will be explored below. In part, it is motivated by the large set of task domains within which determination of an appropriate reference trajectory may involve unnecessary work, or even be effectively impossible. Typical tracking algorithms require the availability of velocity and acceleration reference information: in practice, differentiating

noisy signals is impossible; such schemes are not applicable to tracking unknown time-varying signals. Moreover, evidence mounts that the computational effort required to encode typical static tasks (such as moving in a cluttered space) in terms of exact reference trajectories may be prohibitive (28,29). Finally, since many tasks involve interacting forces and motions between the robot and its environment, a task-encoding methodology limited to the production of reference trajectories may be entirely inoperable. In summary, encoding a task in dynamic terms unrelated to those characterizing the robot or its environment may not be viable.

For the purposes of this entry, an *objective function*,  $\varepsilon: \mathcal{W} \rightarrow \bar{\mathbb{R}}^+$  is a non-negative scalar-valued map on  $\mathcal{W}$  that has isolated critical points. Its associated *gradient vector field* is given by the system

$$\dot{w} = -D_w \varepsilon^T(w)$$

and the resulting trajectory through any initial condition will be called a *fall line* of the system. It can be shown that fall lines are perpendicular to the level surfaces of  $\varepsilon$  (30). The equilibrium states of the gradient system are exactly the critical points—i.e., the extrema—of  $\varepsilon$ ; and since the linearized vector field is symmetric, an equilibrium state of the gradient system is either a source, a sink, or a saddle depending on whether it is a local maximum, minimum, or saddle point of the objective function,  $\varepsilon$ . Thus, gradient systems display very simple dynamic behavior. It will be shown that certain gradient behavior may be duplicated, at least asymptotically, by appropriately compensated Hamiltonian systems and, hence, that such gradients are a particularly convenient feedback structure for robot control.

**Task Encoding Via Objective Functions.** Evidently, tasks within the domain of set-point regulation—reaching and remaining at some desired end point,  $w_d$ —may be encoded as the objective of minimizing

$$\varepsilon \triangleq [w - w_d]^T [w - w_d] \quad (15)$$

The desired end point is the globally asymptotically stable unique equilibrium state of the associated gradient system.

Conversely, by designing a cost function with an isolated global maximum at some undesired Cartesian position, a gradient system may be constructed whose fall lines, from any initial condition different from that point, define motion away from it. This specifies the task of avoiding the undesired position. In the event that there are several obstacles in the work space, each of relatively small physical extent, cost functions attaining an isolated global maximum at the centroid of each obstacle may be summed, and the resulting vector field will specify motions that avoid all of them. A plausible form for such cost functions is the familiar Newtonian potential, which varies in the inverse square of the distance from the obstacles. This and related methodologies have been suggested independently by workers in Japan (31), the Soviet Union (32), and the United States (33). In particular, Khatib has developed a rather general procedure for defining obstacle-avoidance potentials for arbitrary rigid bodies (33). It is not clear, however, that the computational complexity of this procedure makes it any more attractive than the algorithms developed for generating reference trajectories that solve piano-mover-type problems (29).

**Task Encoding Via Gradient Dynamics.** Task domains involving curve tracing may be specified by fall lines of gradient vector fields. Suppose it is desired to reach  $w_d$  via some parameterized curve,

$$c(\zeta) \triangleq \begin{bmatrix} \zeta \\ c_2(\zeta) \\ c_3(\zeta) \end{bmatrix}$$

where  $w_d = c(0)$ . Assume, to avoid technical details, that only the Cartesian position components are considered: errors in orientation may be measured similarly, although the justification requires more discussion. Then the *shaping function*

$$\varepsilon \triangleq w_1^2 + \alpha_2[w_2 - c_2(w_1)]^2 + \alpha_3[w_3 - c_3(w_1)]^2$$

gives rise to a gradient system for which  $w_d$  is again the globally asymptotically stable unique equilibrium state, and whose fall lines “hug” the curve  $c$  more or less sharply depending on the magnitudes of  $\alpha_1, \alpha_2$ .

Fundamental work by Hogan (34) advances persuasive arguments for encoding general manipulation tasks in the form of “impedances.” Impedances and admittances are formal relationships between the force exerted on the world at some Cartesian position and the motion variables—displacement, velocity, acceleration, etc.—at that position with respect to some reference point [or “virtual position” in Hogan’s terminology (34)]. He argues that for purposes of modeling manipulation tasks, the kinematic and dynamic properties of a robot’s contacted environment must be understood as admittances, systems for which the relationship operates as a function describing a specified displacement for any input force. Arguing, further, that physical systems may only be coupled via port relationships that match admittances to impedances and that robots can violate physics no more than any other objects with mass, he arrives at the conclusion that the most general model of manipulation is the specification of an impedance, a system that returns force as a function of motion. By construing motions relative to a virtual position as defining tangent vectors at that position, Hogan notes that an impedance may be defined in terms of a scalar-valued function on the cross product of two copies of the tangent space at each virtual position whose gradient covector determines the relationship between motion and force. Thus, an impedance may be reinterpreted as the gradient covector field of an “objective function,” whose fall lines specify the desired dynamic response of the robot end effector in response to infinitesimal motions imposed by the world. In this context, unlike the other gradient vector field task definitions, it is intended a priori that the dynamics be second order, i.e., define changes of velocity (force) rather than changes of position.

To conclude this brief discussion of task encoding via objective functions or their gradients, it is worth noting that the gradient is a linear operator, and hence combinations of tasks already encoded by means of objective functions are easily specified by appropriately weighted sums. For instance, if it is desired to shape an arm motion around a specified curve while simultaneously avoiding a set of known obstacles, the shaping function may be summed with the avoidance cost functions for each particular obstacle, and the gradient of the sum will preserve the local properties of each. In such cases, however, depending on the nature of the individual objective functions, their sum may define a gradient with unintended stable or partially stable critical points. Thus, globally, the fall lines of

complex gradients may specify “stall” behavior at undesired equilibrium states.

**More General Feedback Structures.** It is certainly possible to imagine the desirability of “first-order dynamic behavior” more complex than can be encoded via the gradient vector field of an objective function. For instance, it has been proposed within the neurobiology community to model phenomena such as animal gait in terms of a dynamic system possessed of a stable limit cycle (35). It would seem equally attractive to use such models as task-encoding feedback structures for robot activities that require repetitive motion. Unfortunately, although “useful” gradient vector field behavior may be embedded “asymptotically” in dissipative Hamiltonian systems, as will be demonstrated in the next section, it is not clear how to do the same for more general dynamics.

A second failing of the objective function methodology is its intrinsic time-invariant character. Even if  $\varepsilon(w, t)$  defines a good objective for each  $t$ , it is not clear how to proceed except in the quadratic case. For example, let  $w_d(t)$  describe the trajectory of a fly the robot should swat as encoded via the objective

$$\varepsilon(w, t) \triangleq [w - w_d]^T [w - w_d]$$

Then the “gradient system”

$$\dot{w} = -w + w_d(t)$$

is a forced, exponentially stable, linear system, and further information concerning the nature of  $w_d$  may afford statements concerning the “steady-state error.” In the special case that  $w_d$  is, itself, the output of some linear system, according to the internal model principle, a higher order linear dynamic compensator may be added to the gradient system with the assurance of asymptotic tracking. The question remains, again, as to how to transplant the internal model principle to the second-order Hamiltonian setting.

**Gradient Vector Fields and Hamiltonian Systems.** It is well known that Lagrangian mechanics may be placed within the more general framework of Hamiltonian dynamic (17). To form the Hamiltonian, we return to the scalar energy functions  $\lambda, \kappa$ , and  $v$  and define the *generalized momenta*,

$$p \triangleq D_{\dot{q}}\lambda^T$$

and the *Hamiltonian*

$$\eta \triangleq p^T \dot{q} - \lambda$$

It is not hard to show that the *Hamiltonian dynamic system*

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & -I \\ I & 0 \end{bmatrix} [D\eta]^T$$

is equivalent to the rigid-body model (Eq. 12), with all external forces and torques set to zero,  $\tau = 0$  (17). Moreover, it is easy to see that any trajectory of this system satisfies  $\eta = \eta_0$ , a constant since

$$\dot{\eta} = D\eta \begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = D\eta \begin{bmatrix} 0 & -I \\ I & 0 \end{bmatrix} [D\eta]^T = 0$$

Finally, note that when the potential energy is free of generalized velocity,  $D_{\dot{q}}\lambda = D_{\dot{q}}\kappa$ , and when, additionally, the kinetic energy is quadratic in velocity,

$$p^T \dot{q} = D_{\dot{q}}\kappa \dot{q} = 2\kappa$$



and, hence, the Hamiltonian represents the total energy of the system,

$$\eta = 2\kappa - \kappa + v = \kappa + v$$

It should be apparent from the derivation of Dynamics that the robot energy terms satisfy these conditions.

By taking this slightly more general perspective, we are able to again use the total energy as a Lyapunov function obtaining a rather simple generalization of the stabilizing PD controller.

**Stability of dissipative Hamiltonian Systems.** The central result presented in this section has been known for at least a century: Lagrange demonstrated the stability of motion around the equilibrium state of a conservative system in 1788 (36); asymptotic stability resulting from the introduction of dissipative forces to a conservative system was discussed by Lord Kelvin in 1886 (37). Over the years these ideas seem to have been rediscovered several times by different engineering communities. For instance, a similar set of observations was made in the context of satellite control in 1966 (38). Credit for first introducing these ideas to the general robotics literature would appear to be due Arimoto and colleagues (39). Similar independent work has appeared more recently by Van der Schaft (40) and this author (41).

It has been shown above that for a broad range of mechanical systems, including actuated kinematic chains, the Hamiltonian is an exact expression for total energy. In a conservative force field this scalar function is a constant (defines a first integral of the equations of motion), and in the presence of the proper dissipative terms it must decay (42). By replacing the gravitational potential term in the energy function with the objective function that defines a task and construing the resulting total energy as a Lyapunov function for the closed-loop robot, set-point regulation may be achieved as follows. Let the input be defined as

$$u \triangleq k(q) + K_2 x_2 + K_1(x_1 - q_d) \quad (16)$$

**Theorem 1.** Let  $\mathcal{J}$  be a simply connected subset of  $\mathbf{R}^n$ . The closed-loop system of Eq. 12, under the state feedback algorithm (Eq. 16),

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -M^{-1}[(B + K_2)x_2 + K_1(x_1 - q_d)]$$

is globally asymptotically stable with respect to the state  $(q_d, 0)$  for any positive definite symmetric matrices  $K_1, K_2$ .

*Proof.* The Lyapunov function

$$v \triangleq \frac{1}{2}[x_1^T K_1 x_1 + x_2^T M(x_1) x_2]$$

has time derivative

$$\dot{v} = x_1^T K_1 x_2 - x_2^T [(B + K_2)x_2 + K_1 x_1] + \frac{1}{2} x_2^T \dot{M} x_2$$

and since  $x_2^T [\frac{1}{2} \dot{M} - B] x_2 = 0$ , as shown in Ref. 42, this evaluates to

$$\dot{v} = -x_2^T K_2 x_2 \leq 0$$

According to LaSalle's invariance principle, the attracting set is the largest invariant set contained in  $\{(x_1, x_2) \in \mathcal{P} : \dot{v} = 0\}$ , which, evidently, is the origin since the vector field is oriented away from  $\{x_2 = 0\}$  everywhere else on that hyperplane.

Note that this control law requires the exact cancellation of any gravitational disturbance. Although  $k(q)$  has a much sim-

pler structure than the moment of inertia matrix,  $M(q)$ , or the coriolis matrix,  $B(q, \dot{q})$ , exact knowledge of the plant and load dynamic parameters would still be required, in general, to permit its computation. Since the dynamic parameters enter linearly in  $k$ , some progress has been made in the design of "adaptive gravity cancellation" algorithms (43) as will be discussed below. A successful adaptive version of this algorithm would remove the need for any a priori information concerning the dynamic parameters.

Note, as well, that with diagonal feedback gains  $K_1, K_2$  and in the absence of gravitational cancellation, the feedback algorithm (Eq. 16) is exactly identical to  $n$  decoupled PD controllers operating at each joint independently, the procedure employed by almost all commercially marketed arms.

**Integrating Gradient Systems by Means of Dissipative Hamiltonian Systems.** The real utility of dissipative Hamiltonian systems in the present context arises from the possibility of embedding a first-order gradient system—the task definition—in the second-order robot arm dynamics with no change in limiting behavior. Let the objective function  $\varepsilon: \mathcal{W} \rightarrow \mathbf{R}$  be defined according to some description in task space, as described above, and let  $\tilde{\varepsilon} \triangleq \varepsilon \circ g$  be the composition of this objective with the kinematics map. The encoded description now takes the form of a gradient system over joint space,

$$\dot{q} = -D\tilde{\varepsilon}(q)^T \quad (17)$$

among whose equilibrium states are desired end points and whose fall lines define a desirable spatial curve or mechanical response function. The desired performance might be simulated on any analog computer with programmable first-order integrators. Instead, it is appealing (and correct) to think of "solving" this gradient system on the "programmable" second-order integrators defined by the intrinsic dynamics of a robot arm.

Among the extrema of  $\varepsilon$ ,

$$\mathcal{E} \triangleq \{w \in \mathcal{W} : D_w \varepsilon = 0\}$$

are the desired task space positions,  $\mathcal{J}$ , the optimal points of the objective function. The task is accomplished if joint space variables tend toward the inverse image of this set

$$\mathcal{S} \triangleq \{q \in \mathcal{J} : g(q) \in \mathcal{E}\}$$

along the trajectories determined by the fall lines of  $\varepsilon$ . Now define the feedback control structure

$$u \triangleq k(q) - K_2 \dot{q} - D_q \tilde{\varepsilon}(q)^T$$

To fix the idea, consider the simple example introduced above of end-point control defined in task space. Based on the error-minimizing objective defined in Eq. 15, the required feedback control law is

$$u = k(q) + K_2 \dot{q} + Dg^T[g(q) - y_d] \quad (18)$$

resulting in convergence from any initial position and velocity toward critical points of  $\tilde{\varepsilon}$ . Explicitly, the closed-loop system of Eq. 12, under the state feedback algorithm (Eq. 18),

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -M^{-1}[(B + K_2)x_2 + D_{x_1} \varepsilon(x_1)^T]$$

has a globally attracting set defined by the critical points

$$\mathcal{E} = \{(q, 0) \in \mathcal{P} : D_q \tilde{\varepsilon}(q)^T = 0\}$$

for any positive definite symmetric matrices  $K_1, K_2$ .

This may be seen as follows. The non-negative Lyapunov function

$$v \triangleq \tilde{\varepsilon}(x_1) + \frac{1}{2}x_2^T M(x_1)x_2$$

has time derivative

$$\dot{v} = (D_{x_1}\tilde{\varepsilon})x_2 - x_2^T[(B + K_2)x_2 + D_{x_1}\tilde{\varepsilon}^T] = -x_2^T K_2 x_2 \leq 0$$

as in the proof of Theorem 1. According to LaSalle's invariance principle, the attracting set is the largest invariant set contained in  $\{(x_1, x_2) \in \mathcal{P} : \dot{v} \equiv 0\}$ , which, evidently, is the set of equilibria,  $\tilde{\mathcal{E}}$ , as claimed.

This result shows that local convergence and global boundedness are assured but that a characterization of the stability properties of individual equilibrium points may be complicated. Namely, if  $\tilde{\varepsilon}$  has critical points outside of  $\mathcal{P}$ , it must be shown that these are not locally attracting equilibrium states of the closed-loop system in order to guarantee global convergence to the desired optima. Expressing the objective function gradient as  $D_q\tilde{\varepsilon} = D_y\varepsilon D_q q$  affords the equivalent formulation of the set of equilibria of the closed loop

$$\tilde{\mathcal{E}} = \{(q, 0) \in \mathcal{P} : D_y\varepsilon^T \in \ker D_q q^T\}$$

This makes clear the two distinct causes of such stall points: local extrema and saddle points of the task space objective,  $\varepsilon$ , and critical points of the output map,  $g$ , i.e., the set of kinematic singularities.

As discussed above, stall behavior due to excess critical points of  $\varepsilon$  is an artifact of the task-encoding methodology. Sophisticated tasks encoded as summed gradients almost inevitably give rise to stall points in  $\mathcal{W}$ . Although it is possible that a more careful construction of  $\varepsilon$  from constituent objectives might mitigate the problem, this is probably an intrinsic limitation in the "global intelligence" of feedback controllers. Research addressing the interplay between higher planning levels and lower control levels should result in guidelines for the degree of supervision required to assure global convergence. For all presently available commercial robots, kinematic singularities may be found in the interior of the work space; thus, the second problem is more intrinsic to an arm and, potentially, of considerable practical concern. For a lucky choice of  $\varepsilon$  it might well turn out that  $\tilde{\mathcal{E}}$  consists only of the points in the solution set; however, this would be unlikely. Some aspects of these questions have been addressed in a recent paper by this author (44).

Of equal pragmatic importance and theoretical interest is a characterization of transient behavior obtaining from those "useful" feedback structures described above. The tasks encoded in terms of gradient dynamics are not achieved merely by asymptotic approach to an extremum of  $\tilde{\varepsilon}$ . Although the theory of linear-time-invariant controllers includes excellent analytical and graphical methods for determining the appropriate magnitude of damping in relation to position gains, as discussed above, no such theory is available for nonlinear Hamiltonian systems. However effective the methodology is in command of steady-state behavior, a gripper that oscillates wildly toward the specified end point is of no practical use. Thus, one of the most important aspects of control research in this project is the study of  $K_2$  relative to  $K_1$  and their nonlinear analogs in Eq. 18. More generally, it would be of great interest to know how to choose a damping function so that the motion of the second-order system projected down onto the zero-velocity plane of  $\mathcal{P}$  follows the fall lines of the original

gradient system as closely as possible: i.e., what is the analogy to a critically damped linear-time-invariant system?

### Servo Control of General Robot Arms

We return here to the paradigm of the servomechanism, wherein tasks are encoded by means of a reference trajectory. For linear-time-invariant dynamics such tracking problems comprise the central arena for the classical control theory discussed above. Unfortunately, there is as yet no generally valid means of translating the full range of these techniques into the nonlinear regime of revolute robot arms. Although the insights of that theory are often applied intuitively with some success—most commercial robots are built with decoupled linear PD servo systems at each joint—theoretical progress has been slow. Ideas that generalize the inverse dynamics approach to tracking are presented below. This approach to robot control, which assumes complete information about the reference trajectory and exact computation of the rigid body model (Eq. 12), is the only tracking paradigm for which global convergence has yet been analytically guaranteed. Then a discussion follows several tracking controllers designed for time-varying linear dynamic systems based on the substitution of its first-order (linear) approximation for the underlying nonlinear model.

It should be noted that there is currently no theoretical understanding of how to generalize the high-grain feedback techniques.

**Exact Linearization by Coordinate Transformation.** In the last decade a significant body of work has developed within the field of nonlinear systems theory concerning the question of when a specified control system has a dynamic structure that is intrinsically linear, or at least, "linearizable." More precisely, presented with a system

$$\dot{x} = f(x, u)$$

it would be of considerable interest to know whether there exists an invertible (memoryless) change of coordinates,

$$\begin{bmatrix} z \\ v \end{bmatrix} = T(x, u)$$

under which the resulting dynamics are linear time invariant,

$$\dot{z} = Az + Bv$$

For, this being the case, the well-understood servo techniques of classical control theory sketched above could be applied to the reference input expressed in the new coordinate system, and the resulting control,  $v_d$ , translated through the inverse coordinate transformation,  $T^{-1}$ , would result in an effective control,  $u_d$ , to be applied to the original system. Early discussion of this question is provided in Refs. 45 and 46, while more recent results have been presented in Ref. 47. More general discussion of this literature may be found in the recent monograph of Isidori (48) or the text by Casti (49).

It will be observed that this policy amounts to exact cancellation of the underlying dynamics of the original system. Thus, as has been remarked, such schemes represent a generalization of the method of pole placement presented above and necessitate exact knowledge of all kinematic and dynamic parameters (including those of the load); a priori knowledge of the reference trajectory; and the ability to compute exactly

and implement through a set of actuators the entire dynamics in real time. Work by a number of researchers, most notably Hollerbach (50), has persuasively demonstrated that such computation is possible in real time, and computational architectures have already been designed to do so (51,52). Recent empirical results (53) suggest that this methodology may achieve good results when the requisite a priori information concerning dynamic parameters and reference trajectory is available.

**Computed-Torque Algorithm.** In the context of the robot equations (Eq. 12) these ideas lead to the technique of "computed torque," which has been proposed independently under a variety of names by several different researchers over the last five years (54–56). It seems most instructive to present the variations in the computed torque algorithm as particular examples of the following exact linearization scheme. Let  $h: \mathcal{J} \rightarrow \mathbf{R}^n$  be a local diffeomorphism. Under the change of coordinates, defined by  $T: \mathcal{P} \times \mathcal{U} \rightarrow \mathbf{R}^{3n}$ ,

$$\begin{bmatrix} z_1 \\ z_2 \\ v \end{bmatrix} = T(x_1, x_2, u) \triangleq \begin{bmatrix} h(x_1) \\ Dh x_2 \\ \dot{D}h x_2 - Dh M^{-1}[B x_2 + k(x_1) - u] \end{bmatrix} \quad (19)$$

system 12 has linear-time-invariant dynamics given by

$$\dot{z}_1 = z_2 \quad \dot{z}_2 = v \quad (20)$$

with output map

$$w = g \circ h^{-1}(z_1)$$

This may be seen according to the definition of  $z_1, z_2$  by applying the chain rule,

$$\dot{z}_1 = Dh x_2 = z_2$$

and by noting

$$\dot{z}_2 = \dot{D}h x_2 - Dh M^{-1}[B x_2 + k - u]$$

This is not the most general class of transformations that might be used to linearize Eq. 12, according to the nonlinear systems literature cited above (e.g., Ref. 47), but it includes methods commonly encountered in the field of robotics. In particular, for nonredundant kinematics, if we identify  $h$ , the first component of  $T$  with the kinematic map,

$$h(x_1) \triangleq g(x_1),$$

then, locally,  $T$  not only linearizes Eq. 12 but also dynamically decouples each input and output pair, e.g., as reported in Refs. 54 or 56, since  $w = z_1$ .

As suggested above, the advantage of this approach is that the servo design problem may now be addressed by the classical methods introduced in Single-Degree-of-Freedom Robot Arm, shifting the problem of task specification to lie within the domain of some independent "higher level" process. Namely, suppose such a higher level algorithm produces a desired trajectory in work space,  $w_d(t)$ , which it is required that the robot reproduce. Defining  $z_d \triangleq [h \circ g^{-1}(w_d), Dh Dg^{-1} \dot{w}_d]^T$ , it is quite straightforward to choose some linear feedback compensator,  $v_{fb} \triangleq Kz$ , and feedforward precompensator,  $v_{pc} \triangleq \Gamma[z_d]$ , that determine a "classical" linear control law,

$$v_d \triangleq -Kz + \Gamma(z_d) \quad (21)$$

under whose action the output of Eq. 20 behaves in a desired fashion with respect to the reference input,  $z_d$ . The particular choice of linear control scheme determines the nature of overall performance along the lines explored. Since the relationship between  $(x, u)$  and  $(z, v)$  has no dynamics (is "memoryless"), the input to the robot (Eq. 12) defined by the inverse coordinate transformation for  $u$  under  $T$  (obtained by solving for  $u$  in the last row of Eq. 19),

$$u_d \triangleq Bx_2 + k(x_1) + MDh^{-1}[v_d - Dh x_2] \quad (22)$$

forces the output  $w(t) \equiv g[h^{-1}(z_1)]$ .

Perhaps the best known example of this approach in the robotics literature is provided by the "resolved acceleration" method of Ref. 55. A control law,  $\Gamma[z_d]$ , is chosen for the linearized system using the "inverse filter" method,

$$\Gamma_{id}(z_d) \triangleq Kz_d + \ddot{z}_{d1} = K \begin{bmatrix} h \circ g^{-1}(w_d) \\ Dh Dg^{-1} \dot{w}_d \end{bmatrix} + \left[ \frac{d}{dt} Dh Dg^{-1} \right] \dot{w}_d + Dh Dg^{-1} \ddot{w}_d$$

i.e., the inverse of the filter specified by the equivalent closed-loop linear-time-invariant system (Eq. 20). The control law,  $u_d$ , to be applied to the robot is then given by Eq. 22. Note that this choice,  $h \equiv g$ , satisfies the conditions for a local diffeomorphism almost everywhere in  $\mathcal{J}$ : the condition fails at the "kinematic singularities,"

$$C \triangleq \{q \in \mathcal{J}: \text{rank}(Dg) < \dim \mathcal{W}\},$$

the critical points of  $g$ . Most realistic robots have kinematic singularities whose image under  $g$  is in the interior of  $\mathcal{W}$  and that may not be easily located; hence, such a transformation may be impracticable.

As an alternative example, if the task is specified as a trajectory in joint space,  $x_d \triangleq [q_d, \dot{q}_d]^T$ , this methodology corresponds to a trivial change of coordinates under the identity map,  $h \equiv I$ , and the control law (Eq. 22) reduces to

$$u_d \triangleq Bx_2 + k(x_1) + Mv_d$$

The experimental results of exact linearization schemes in robotics reported to date (53) have employed this version of the algorithm.

**Robust Cancellation via Sliding Modes.** A number of researchers have explicitly addressed the problems likely to attend schemes that rely on exact cancellation, as mentioned in the introduction of this section. Here, we present an interesting approach based on the methods of "variable-structure systems." The classification of this technique under the rubric of "robust" methods is justified in that the method requires complete knowledge only of the terms in the existing dynamics that may become unbounded.

The theory of "variable-structure systems" has been developed in the Soviet Union over the last two decades (57) and applied to the problem of robot control in recent years (58–60). For purposes of this entry, the approach may be presented in the context of objective function methods as follows. In phase space geometric relationships between state variables imply dynamic behavior, as has been seen in previous sections. Rather than specifying a goal in terms of an output objective function, the sliding mode objective specifies a desired error dynamics:

$$s(x, q_d(t)) \triangleq x_2 - \dot{q}_d + \lambda(x_1 - q_d)$$

which, if zero, implies that the position and velocity errors between the desired and true trajectory are asymptotically approaching zero. It will be noted that the attendant objective function,

$$\varepsilon(x, t) \triangleq s^T s$$

is explicitly time varying; thus, the natural control strategies cannot be applied with confidence. Instead, the existing dynamics are cancelled or forced toward those desired by making systematic use of the rigid-body model (Eq. 11).

Specifically, the acceleration equations of rigid-body-model dynamics (Eq. 12) may be rewritten,

$$\ddot{x}_2 = -M^{-1}[Bx_2 + k(x_1) - \tau] = H(x)p(x) + M^{-1}\tau$$

where  $H(x)$  is an array containing only bounded terms (constants and transcendental functions in the state) and  $p(x)$  is a vector containing in each entry a monomial,  $x_{1i}^k x_{2j}^l$ , consisting of (unbounded) powers in the state variables. Although exact knowledge of the latter is required, the only information concerning the bounded terms takes the form of a superior magnitude for each entry,  $\bar{H}$ , with the property,

$$\bar{H} > \sup_{x \in \mathcal{P}} |H(x)|$$

Letting the control input be

$$u \triangleq M[-(\bar{H}|p(x)|)\text{sgn}(s) + \ddot{q}_d - \lambda(x_2 - \dot{q}_d)]$$

it follows that

$$\begin{aligned} \dot{\varepsilon} &= s^T \dot{s} \\ &= s^T[\dot{x}_2 - \ddot{q}_d + \lambda(x_2 - \dot{q}_d)] \\ &= s^T[(Hp)\text{sgn}^2(s) - (\bar{H}|p|)\text{sgn}(s)] \\ &\leq s^T \text{sgn}(s)[\text{sgn}(s)|H| - \bar{H}]|p| \\ &= -|s|^T h^+(x) < 0 \end{aligned}$$

where all entries in  $h^+$  are positive so that the derivative is always negative. Implicit in these equations is the convention that the absolute value, sign (sgn), and inequality notations applied to arrays are understood to hold component by component and that the product of two vectors denotes their component-by-component (commutative) product. Thus, all trajectories originating away from the objective surface,  $\varepsilon \equiv 0$ , tend toward it asymptotically. Since the vector field resulting from this control law is discontinuous, additional mathematical justification is required (60) in order to infer that the theoretical behavior on the surface is the desired simplified dynamics.

In practice, as has been discussed previously, limitations in accuracy and the finite response time of sensors and actuators preclude the possibility of realizing the specified control discontinuity. In consequence, physical implementations of this method give rise to *chattering*—rapid fluctuations of the state across the objective surface (58). Such high-frequency artifacts are extremely undesirable: excitation of (typically underdamped) unmodeled dynamics is the inevitable result. It may be noticed, moreover, that the success of the method depends on sufficiently large control magnitudes of the appropriate sign. Thus, exact knowledge of  $M$  is required to preserve sign information,  $|s|$ , through the input coupling, and it is likely that prohibitively large input magnitudes would be specified by conservative dependence on a priori bounds,  $\bar{H}$ .

An interesting modification of this method addressing these

problems has been proposed by Sastry and Slotine (59,60). They replace the discontinuous switching rule,  $|s|$ , with a smoothed version that trades ultimate tracking precision for improved transient response. They investigate, as well, the possibility of abandoning exact cancellation of the input coupling terms,  $M$ , in favor of “sign preservation” through suitable diagonally dominant positive approximations.

**Other Coordinate Transformation Schemes.** Other choices for  $h$  might be imagined: if the kinematics and dynamic parameters that give rise to system 11 define a moment of inertia matrix whose square root is the Jacobian of some map, a much simpler coordinate transformation,  $T$ , results. This has been explored by the author in Ref. 61. For ease of discussion, define the set of *square roots* of a smooth positive definite symmetric matrix valued function,  $M(q)$ , as

$$\mathcal{N}(M) \triangleq \{N \in C^\infty[\mathcal{J}, \mathbf{R}^{n \times n}]: NN^T = M\}$$

Note that since  $M$  is assumed to be positive definite,  $\mathcal{N}(M)$  is not empty, and any  $h$  satisfying the hypothesis is an immersion.

Suppose there exists a smooth map  $h: \mathcal{J} \rightarrow \mathbf{R}^n$  such that  $Dh^T = N \in \mathcal{N}(M)$ . Then under the change of coordinates, defined by  $T: \mathcal{P} \times \mathcal{Q} \rightarrow \mathbf{R}^{3n}$ ,

$$\begin{bmatrix} z_1 \\ z_2 \\ v \end{bmatrix} = T(x_1, x_2, u) \triangleq \begin{bmatrix} h(x_1) \\ N^T x_2 \\ N^{-1}[u - k(x_1)] \end{bmatrix} \quad (23)$$

the system has linear-time-invariant dynamics given by Eq. 20. To show this, note that  $z_2 \triangleq Dh x_2 = \dot{z}_1$ . Moreover,

$$\begin{aligned} \dot{z}_2 &= D\dot{h}x_2 + Dh\dot{x}_2 \\ &= \dot{N}^T x_2 - N^T[NN^T]^{-1}[Bx_2 + k(x_1) - u] \quad \text{from Eq. 12} \\ &= [\dot{N}^T - N^{-1}B]x_2 + N^{-1}u \end{aligned}$$

and it remains to show that  $[\dot{N}^T - N^{-1}B] = 0$ . To see this,

$$\begin{aligned} Bx_2 &\triangleq \dot{M}x_2 - \frac{1}{2}[D_q z_2^T z_2]^T \\ &= [\dot{N}N^T + N\dot{N}^T]x_2 - \left[D_q \frac{d}{dt} z_1\right]^T z_2 \\ &= [\dot{N}N^T + N\dot{N}^T]x_2 - \left[\frac{d}{dt} D_q z_1\right]^T z_2 \\ &= N\dot{N}^T x_2 \end{aligned}$$

from which the result follows. Note that the exchanged order of differentiation in the third line is justified since  $z_1$  is continuously differentiable in both  $q$  and  $t$ .

Some of the advantages of a coordinate transformation based on the square root of the moment of inertia matrix are immediately evident. Given the choice of classical controller,  $v_d$ , from Eq. 21, the inverse transformation for  $u$  in terms of  $z$ ,  $v$  is considerably simplified,

$$u_d \triangleq Dh^T v_d + k(x_1)$$

in comparison to Eq. 22. Moreover, since  $h$  is an immersion,  $T$  may be computed everywhere on  $\mathcal{J}$ . The conditions for the existence of such a map,  $h$ , whose Jacobian is in  $\mathcal{N}(M)$ , were given by Riemann in 1854 (62) and amount to the question of when an apparently non-Euclidean metric is “flat”—i.e., gives rise to a space of zero curvature. The transformation in question is a particular instance of a local *isometry*, and a more general problem of some interest concerns the existence of

other isometries that simplify the dynamics of Eq. 12. Research exploring whether any useful class of robot arms gives rise to a flat metric and whether more general isometries might be helpful for purposes of control continues (61).

**Linear Approximations.** Since the equations of motion of a robot are analytic in the state variables, it follows that all partial derivatives,  $A(x, u) \triangleq dx f, B(x, u) \triangleq du f$ , exist and are analytic as well. Suppose that  $x(t), x'(t)$  are two motions governed by Eq. 12, resulting from the application of the control inputs  $u(t), u'(t)$ , respectively, starting from the same initial conditions. If their differences,  $\tilde{x} \triangleq x' - x; \tilde{u} \triangleq u' - u$ , are small throughout the time interval of interest, according to Taylor's formula,  $f(x', u') \approx f(x, u) + A(x, u)\tilde{x} + B(x, u)\tilde{u}$ . Since  $\dot{\tilde{x}} = f(x', u') - f(x, u)$ , it follows that the evolution of  $\tilde{x}$  is governed approximately by  $\tilde{u}$  according to the dynamic law

$$\dot{\tilde{x}} = A(x(t), u(t))\tilde{x} + B(x(t), u(t))\tilde{u}$$

This is a time-varying linear differential equation, and it might be anticipated that some of the insights explored above carry over to such a system more readily than to the actual robot equations (Eq. 12). In point of fact, while a formal expression for the I/O properties of such a system may be readily exhibited, its analytical properties (much less its explicit computation) may not be any better understood than the original nonlinear system from which it has been derived. However, since these functions are all smooth, the linear coefficients are approximately constant over sufficiently small intervals of time: if control action is possible on such rapid time scales, the insights of the section on single-degree-of-freedom robot arms may be of some use.

**Gain Scheduling.** Suppose a control  $u_d$  has been devised off-line in such a fashion that the resulting trajectory  $x_d$  exhibits desired properties. For example,  $u_d$  might be chosen according to the ideal computed torque method discussed above; it might instead be the result of some iterative off-line optimization procedure. The relevant point is that off-line numerical simulation of system 12 for specified values of  $u$ , while computationally costly, is a straightforward procedure. In turn, the availability of  $u_d(t), x_d(t)$  affords off-line computation of the Jacobian matrices  $A, B$  along that motion. According to the reasoning stated above, if  $u_d$  is applied to the true system, at any instant of time  $t_0$  small deviations  $\tilde{x}$  around  $x_d$  are governed by the instantaneous linear-time-invariant dynamics

$$\dot{\tilde{x}} = A_0\tilde{x} + B_0\tilde{u}$$

where  $A_0 \triangleq A(x_d(t_0), u_d(t_0)), B_0 \triangleq B(x_d(t_0), u_d(t_0))$ . Over an interval around  $t_0$  small enough to admit these approximations, then, a linear-time-invariant feedback controller can be constructed with gains set according to the well-understood design principles sketched in the beginning of this entry. At some later time  $t_1$  a different set of gains to achieve the same purpose will be required to compensate for the dynamics caused by  $A_1, B_1$ . The technique of assigning a different set of linear-time-invariant compensator gains to different regimes of system behavior is widely known as "gain scheduling."

The chief objection to this method is that stable compensation is guaranteed only in some local neighborhood of the nominal trajectory. If excessive disturbances are present—whether due to inaccuracies in the model, in its computation, or as a result of unexpected contingencies in the environ-

ment—it is possible that tracking will fail badly: the true system response might even become unbounded (i.e., saturate). Another practical cause for degraded performance is a consequence of the continuous evolution of the Jacobian coefficients over the time interval that the compensator is assigned a set of constant gains according to schedule. This might be ameliorated by increasing the scheduling rate, but it is possible that the time variation of the linearized model, governed by the rate at which the reference signal changes as well as the sensitivity of the linearized model to changing nominal operating points, could prove simply too fast for the controller.

The technique of local linearization is frequently encountered in the robotics control literature, and it is worth noting a few representative examples. A study of control issues arising in the context of the linearized dynamic model around specific fixed positions in the work space is given in Ref. 63. A computer-automated environment for designing controllers based on linearization around the "computed torque" trajectory is described in Ref. 64. An optimization method is described in Ref. 65 for solving obstacle-avoidance problems which returns a numerically computed nominal input-response pair that might be used as the basis for a local linearization feedback controller.

**Local Adaptive and Learning Techniques.** If the local behavior of a nonlinear system is essentially governed by shifting linear-time-invariant dynamics in the sense explored in the preceding section, it seems fruitful to inquire regarding the existence of a single compensator that would suffice to control all the disparate linear-time-invariant systems over that range. Alternatively, if a linear compensator exists that adequately controls the locally linearized plant at every instant of time along some trajectory, it might be possible to build a time-varying compensator that adjusts its parameters accordingly—an "automatic gain scheduler." The first of these ideas leads to the notion of robust nonlinear controllers. This topic properly deserves a separate section here. Unfortunately, although most commercial robots are built using the linear "PD" techniques, there is very little understanding of how these ideas should generalize to the nonlinear setting, and, accordingly, no material to fill such a section here.

The second idea leads to the notion of adaptive control, which is treated here in its "local" form. As shown in the previous section, the nonlinear dynamics of the robot arm at any point in the work space are locally characterized by an instantaneous linear-time-invariant vector field. If a standard adaptive algorithm converges quickly enough, it is plausible that adaptive estimates of the current instantaneous linearized coefficients will result in an effective control. Thus, theoretical questions introduced by such algorithms reduce to the study of the sort of schemes presented in Adaptive Control. Implementation of these ideas has been considered, e.g., in Refs. 66–68. An interesting hybrid on-line open- off-line closed-loop local adaptive strategy for robot arms has been proposed in recent years and is worth independent mention.

The original proponent of such a scheme, which uses a sequence of open-loop control inputs to force the output to track a desired signal, seems to have been Arimoto (69). The error between the true and desired response resulting from a previous control input is used to modify that signal and produce the next control input in the sequence. The entire time history of the error function resulting from a particular iterate is used in the construction of the next. Explicitly, suppose a con-

trol input,  $u_i(t)$ , has been chosen to track the desired output,  $y_d(t)$ , starting from initial conditions  $x^*$  with resulting velocity errors

$$e_i(t) \triangleq \dot{q}_d(t) - x_2(t)$$

The simplest version of the scheme calls for the next control candidate to be defined by

$$u_{i+1}(t) \triangleq u_i(t) + \Phi e_i(t)$$

and applied from the same initial condition,  $x^*$ . When the underlying plant dynamics are linear, Arimoto shows that the error sequence does indeed converge uniformly for arbitrary initial conditions  $x^* \in \mathcal{P}$ . The same convergence result may be applied locally—i.e., by assuming that no matter how bad the intermediate behavior, the state variables remain within some compact domain in  $\mathcal{P}$ —to the nonlinear robot dynamics of Eq. 11.

**Global Adaptive Controllers.** It has been mentioned above that a rigorous theory of adaptive control for linear-time-invariant systems is a relatively recent development. Accordingly, the prospects for theoretically sound adaptation algorithms for general nonlinear systems seems quite remote in the near future. Fortunately, in contrast, the possibility of developing practicable globally stable adaptive robot control algorithms within the next few years seems quite bright. This optimism is grounded on the following observations.

The rigid-body model (Eq. 11) is highly nonlinear in the state  $x$  and kinematic parameters but linear in the dynamic parameters (as will be verified shortly). Future robots will probably be built using the direct-drive technology (70). This implies that the omissions in rigid-body model take the form presented in Additional Dynamics which is still linear in (an augmented set of) the dynamic parameters in contrast to the additional nonlinearities presented in Local Nonlinearities. Adaptive problems with linear parameter dependence are much more tractable than general nonlinear problems, as will be seen below.

To see that the system of Eq. 11 is linear in the dynamic parameters, note that

$$M(q) = \sum_{i=1}^n [\tilde{H}_i(q)]^T \tilde{P}_i \tilde{H}_i(q)$$

where  $\tilde{H}_i(q) \triangleq D_q F_i^S$  depends entirely on joint positions and kinematic parameters, and

$$\tilde{P}_i \triangleq \begin{bmatrix} \bar{J}_i & \bar{\mu}_i p_i \\ \bar{\mu}_i p_i^T & \bar{\mu}_i \end{bmatrix} \otimes I$$

depends on the dynamic parameters. Since

$$([\tilde{H}_i(q)]^T \tilde{P}_i \tilde{H}_i(q))^S = ([\tilde{H}_i(q)]^T \otimes [\tilde{H}_i(q)]^T) \tilde{P}_i^S$$

(refer to Appendix A for material concerning the “stack representation”), defining

$$\tilde{H}(q) \triangleq \begin{bmatrix} ([\tilde{H}_1(q)]^T \otimes [\tilde{H}_1(q)]^T) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & ([\tilde{H}_n(q)]^T \otimes [\tilde{H}_n(q)]^T) \end{bmatrix}$$

$$p \triangleq \begin{bmatrix} \tilde{P}_1^S \\ \vdots \\ \tilde{P}_n^S \end{bmatrix}$$

yields  $M(q)^S = H(q)p$ . Since  $M$  is linear in  $p$ , its derivatives must be as well, and hence,  $B(q, \dot{q})\dot{q} = H'p$ . It is clear from the derivation in Dynamics that  $k$  is linear in  $p$ ,  $k(q) \triangleq H''(q)p$ .

**Adaptive Computed Torque.** Now consider the general problem of adaptive control for the robot servo problem. Suppose a desired trajectory,  $q_d$ , is given, along with a linear precompensating scheme,  $u_{pc} \triangleq \Gamma[q_d]$ , which makes  $x_{m1}$ , the output of a forced model reference dynamic system,

$$\dot{x}_m = \begin{bmatrix} 0 & I \\ -K_1 & -K_2 \end{bmatrix} x_m + \begin{bmatrix} 0 \\ I \end{bmatrix} u_{pc}$$

track  $q_d$  in an acceptable fashion. Assume, moreover, that the structure of the robot dynamics  $H, H', H''$  is entirely known although the dynamic parameters  $p$  are not.

If  $p$  were known, the appropriate control strategy would be a generalization of the pole placement scheme,

$$\begin{aligned} u_d &\triangleq k(q) + B(q, \dot{q})\dot{q} + M[-K_1\dot{q} - K_2\ddot{q} + u_{pc}] \\ &= H''(q)p + H'p + ([ -K_1\dot{q} - K_2\ddot{q} + u_{pc}]^T \otimes I)Hp \\ &= \tilde{H}(q, \dot{q}, u_{pc})p, \end{aligned}$$

since this “linearizes” the robot dynamics in the sense of exact linearization by coordinate transformation and places the poles such that the closed-loop system has the dynamics of the reference model. Thus a generalization of the reasoning in Adaptive Control suggests that the appropriate adaptive control take the form

$$u_{ad} \triangleq \tilde{H}\hat{p} = u_d + \tilde{H}[\hat{p} - p]$$

where  $\hat{p}$ , the parameter estimate, will be continuously adjusted over the course of the robot's motion. The closed-loop error equations,  $e \triangleq x_m - x$ , under this control take the form

$$\dot{e} = \begin{bmatrix} 0 & I \\ -K_1 & -K_2 \end{bmatrix} e + \begin{bmatrix} 0 \\ I \end{bmatrix} (M^{-1}\tilde{H}[\hat{p} - p])$$

Again generalizing from the earlier section on adaptive control, the adaptive law should depend on a Lyapunov function for the reference plant. A convenient choice is  $v \triangleq x_m^T P_m x_m$ , where

$$P_m \triangleq \begin{bmatrix} K_1 & \frac{1}{2}K_2 \\ \frac{1}{2}K_2 & I \end{bmatrix}$$

may be shown to be positive definite as long as the reference system is chosen such that  $4K_1 - K_2^2$  is positive definite. Note that  $\dot{v} = -x_m^T(K_2 \otimes I)P_m x_m$  along the motion of the reference system, and this is easily seen to be negative definite under the further assumption that  $K_1, K_2$  commute. Given these assumptions, the choice of adaptive law consonant should be

$$\dot{\hat{p}} = \tilde{H}^T M^{-1}(\frac{1}{2}K_2 e_1 + e_2)$$

Unfortunately, this law is entirely impracticable since by involving  $M$  explicitly, it presupposes the availability of the very information that necessitated an adaptive approach in the first place.

It is quite appealing to consider instead the adaptive law,

$$\dot{\hat{p}} = \tilde{H}^T \hat{M}^{-1}(\frac{1}{2}K_2 e_1 + e_2)$$

where  $\hat{M}$  is computed according to the recipe for  $M$  using the current value of the parameter estimate,  $\hat{H}\hat{p}$ . Unfortunately, while  $M$  is known to be positive definite and, therefore, invertible over all  $q \in \mathcal{J}$ , the estimate at any given instant,  $\hat{M}$ , does



not enjoy such a guarantee. Even if this condition could be assured, it is no longer obvious how to demonstrate convergence of the overall scheme.

Interesting recent work by Craig, Hsu, and Sastry, (71) presents an approach to this problem based on the inverse dynamics precompensator,

$$u_{pc} \triangleq \ddot{q}_d + K_2 \dot{q}_d + K_1 q_d$$

Their analysis examines a "reverse causal" precompensator-robot forward path system—i.e., the reference model driven with inverse dynamics involving the true robot position and velocity. This leads to error equations of the form

$$\dot{e} = \begin{bmatrix} 0 & I \\ -K_1 & -K_2 \end{bmatrix} e + \begin{bmatrix} 0 \\ I \end{bmatrix} (\hat{M}^{-1} \bar{H}^* [\hat{p} - p])$$

and an adaptive law of the form

$$\dot{\hat{p}} = \bar{H}^* \hat{M}^{-1} (\frac{1}{2} K_2 e_1 + e_2)$$

where  $\hat{M}$  is prevented from becoming singular or unbounded by explicitly arresting the adaptation when  $\hat{p}$  leaves a predetermined compact region in the positive orthant of parameter space. Convergence obtains after a finite number of "adaptation resets." Unfortunately, as  $\bar{H}$  contains reference trajectory acceleration terms, so does  $\bar{H}^*$ —a portion of the adaptive law to be synthesized on-line—contain true response acceleration terms  $\ddot{q}$  that would require either use of accelerometers or instantaneous differentiation of real-time signals in any physical implementation.

**Adaptive Gravity Cancellation for a PD Controller.** Since the complete paradigm of adaptive control does not easily translate into the nonlinear robotic setting, it is sensible to consider schemes where adaptation plays a reduced role. Assume that the desired task is achieved by a natural control law of the kind examined above, and consider the problem of adaptive cancellation of the gravity term mentioned in Stability of Dissipative Hamiltonian Systems.

Specifically, it is required to cancel the destabilizing portion of the vector field,

$$k(q) = H\bar{p}$$

The appropriate control input is given as

$$u_{ad} \triangleq H - r\bar{p} - K_2 \dot{x}_2 - K_1 x_1$$

with  $x_1 \triangleq q_d - q$ , and  $\bar{p}$ , the present estimate of the unknown gains that will be adjusted continuously during the robot's motion. According to the earlier discussion of adaptation, the construction of the adaptive law requires use of an extended Lyapunov function. Unfortunately, the only presently available candidate is the total energy function,

$$v \triangleq \frac{1}{2} [x_2^T M x_2 + x_1^T K_1 x_1]$$

whose time derivative along trajectories of the (perfectly gravitationally canceled) closed-loop system was shown to be negative semidefinite rather than negative definite. Proceeding anyway, in analogy to that discussion, set the adaptive law to be

$$\dot{\hat{p}} = -H^T x_2$$

Notice that this is a practicable procedure since all explicit dependence on  $p$  is canceled. The closed-loop behavior is then governed by the equation

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = -M^{-1}[(B + K_2)x_2 + K_1(x_1) + H\bar{p}] \quad (24)$$

$$\dot{\hat{p}} = H^T x_2$$

It is shown in Ref. 43 that this system has a stable origin and gives rise to bounded solutions whose limit set is contained in the subspace

$$\mathcal{L} \triangleq \left\{ \begin{bmatrix} x \\ \bar{p} \end{bmatrix} : x_2 = 0 \right\}$$

thus, each physical trajectory will converge to some spatial position  $q_0 \in \mathcal{J}$ , and the parameter estimate  $\hat{p}$  will converge to some constant  $\hat{p}_0 \in \mathbf{R}^{10n}$ . Unfortunately, the result says nothing about the relation of these constants to their desired values. In fact, the most likely result of this procedure would be entirely unsatisfactory. For all those positions  $q_d \in \mathcal{J}$  at which  $H(q_d)$  has full rank, the origin of the system of Eq. 24 lies in the interior of a smooth submanifold of  $\mathcal{L}$  specified by

$$\mathcal{M} \triangleq \left\{ \begin{bmatrix} q \\ 0 \\ \bar{p} \end{bmatrix} : \bar{p} \in H^{-1}K_1(q_d - q) \right\}$$

which is a set of equilibrium states. Thus, not only is the origin nonattracting but also solutions will converge to constants in  $\mathcal{M}$  however distant from the origin that manifold extends. Physically, this corresponds to a command torque based on a spatial error whose corruption by the parameter error exactly balances the gravitational force vector at a particular point in  $\mathcal{W}$ . Research attempting to improve this result is currently in progress.

## Appendix: Some Notation

If  $f: \mathbf{R}^n \rightarrow \mathbf{R}^m$  has continuous first partial derivatives, denote its  $m \times n$  Jacobian matrix as  $Df$ . When we require only a subset of derivatives, e.g., when  $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ , and we desire the Jacobian of  $f$  with respect to the variables  $x_1 \in \mathbf{R}^{n_1}$ , as  $x_2$  is held fixed, we may write

$$D_{x_1} f \triangleq Df \begin{bmatrix} I_{n_1} \times n_1 \\ 0 \end{bmatrix}$$

If  $A \in \mathbf{R}^{n \times m}$ , the "stack" representation (72) of  $A \in \mathbf{R}^{nm}$  formed by stacking each column below the previous will be denoted  $A^S$ . If  $C \in \mathbf{R}^{p \times q}$ , and  $A$  is as above, the Kronecker product of  $A$  and  $C$  is

$$A \otimes C \triangleq \begin{bmatrix} a_{11}C & \cdots & a_{1m}C \\ a_{21}C & \cdots & a_{2m}C \\ \vdots & \ddots & \vdots \\ a_{n1}C & \cdots & a_{nm}C \end{bmatrix} \in \mathbf{R}^{np \times mq}$$

Finally, if  $B \in \mathbf{R}^{m \times p}$ , and  $A$  and  $C$  are as above, it can be shown that

$$[ABC]^S = (C^T \otimes A)B^S$$

A second useful identity between matrices and their stack representation is

$$\text{tr}\{DK^T\} = (D^S)^T L^S$$

## BIBLIOGRAPHY

1. S. Bennett, *A History of Control Engineering: 1800–1930*, Peregrinus, London, 1979.
2. O. Mayr, *The Origins of Feedback Control*, MIT Press, Cambridge, MA, 1970.
3. R. C. Dorf, *Modern Control Systems*, Addison-Wesley, Reading, MA, 1974.
4. I. Horowitz, "Feedback systems with nonlinear uncertain plants," *Int. J. Contr.* **36**(1), 155–171 (1982).
5. C. T. Chen, *Introduction to Linear System Theory*, Holt, Rinehart and Winston, New York, 1970.
6. C. A. Desoer, *Notes for a Second Course on Linear Systems*, Van Nostrand, New York, 1970.
7. M. A. Arbib, R. E. Kalman, and P. L. Falb, *Topics in Mathematical System Theory*, McGraw-Hill, New York, 1969.
8. V. I. Arnold, *Ordinary Differential Equations*, MIT Press, Cambridge, MA, 1978.
9. J. P. Lasalle and S. Lefschetz, *Stability by Lyapunov's Direct Method with Applications*, Academic, New York, 1961.
10. J. L. Bower and P. M. Schultheiss, *Introduction to the Design of Servomechanisms*, Wiley, New York, 1958.
11. J. G. Truxal, *Automatic Feedback Control System Synthesis*, McGraw-Hill, New York, 1955.
12. K. S. Narendra, Y.-H. Lin, and L. S. Valavani, "Stable adaptive controller design, Part ii: Proof of stability," *AC* **AC-25**(3), 440–448 (June 1980).
13. K. S. Narendra and Y. H. Lin, "Design of stable model reference adaptive controllers," *Applications of Adaptive Control*, Academic, New York, pp. 69–130, 1980.
14. K. S. Narendra (ed.), *Adaptive and Learning Systems: Theory and Applications*, Plenum, New York, 1986.
15. J. P. Lasalle, *The Stability of Dynamical Systems*, Volume 25 of *Regional Conference Series in Applied Mathematics*, SIAM, Philadelphia, PA, 1976.
16. K. S. Narendra and L. S. Valavani, "A comparison of Lyapunov and hyperstability approaches to adaptive control of continuous systems," *IEEE Trans. Autom. Contr.* **AC-25**(2), 243–247 (April 1980).
17. V. I. Arnold, *Mathematical Methods of Classical Mechanics*, Springer-Verlag, New York, 1978.
18. L. Auslander and R. E. MacKenzie, *Introduction to Differentiable Manifolds*, Dover, New York, 1977.
19. J. A. Thorpe, *Elementary Topics in Differential Geometry*, Springer-Verlag, New York, 1979.
20. R. P. Paul, *Robot Manipulators: Mathematics Programming and Control*, MIT Press, Cambridge, MA, 1981.
21. C. S. G. Lee, R. C. Gonzalez, and K. S. Fu, *Tutorial on Robotics*, IEEE Computer Society, New York, 1983.
22. H. Goldstein, *Classical Mechanics*, Addison-Wesley, Reading, MA, 1980.
23. A. K. Bejczy, Robot Arm Dynamics and Control, Technical Report 33-699, Jet Propulsion Laboratory, Pasadena, CA, 1974.
24. L. M. Sweet and M. C. Good, "Redefinition of the robot motion-control problem," *IEEE Contr. Syst. Mag.* **5**(3), 18–25 (August 1985).
25. Electro-Craft Corp, *DC Motors, Speed Controls, Servo Systems*, Electro-Craft, Hopkins, MN, 1980.
26. R. Goor, A New Approach to Minimum Time Robot Control, Research Publication GMR-4869, General Motors Research Laboratories, Warren, MI, November 1984.
27. P. Moroney, *Issues in the Implementation of Digital Feedback Compensators*, MIT Press, Cambridge, MA, 1983.
28. J. Reif, Complexity of the Mover's Problem and Generalizations, *Proceedings of the Twentieth Symposium of the Foundations of Computer Science*, 1979.
29. J. T. Schwartz and S. M., On the Piano Mover's Problem. I, Computer Science Department Technical Report 39, NYU Courant Institute, New York, October 1981.
30. M. W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra*, Academic Press, New York, 1974.
31. F. Miyazaki and S. Arimoto, Sensory Feedback Based on the Artificial Potential for Robots, *Proceedings of the Ninth IFAC*, Budapest, Hungary, 1984.
32. V. V. Pavlov and A. N. Voronin, "The method of potential functions for coding constraints of the external space in an intelligent mobile robot," *Sov. Autom. Contr.* **6**(1984).
33. O. Khatib, "Real time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.* **5**(1), 90–99 (Spring 1986).
34. N. Hogan, "Impedance control: An approach to manipulation," *ASME J. Dynam. Syst. Meas. Contr.* **107**, 1–7 (March 1985).
35. E. Saltzman and J. A. Scott Kelso, Skilled Actions: A Task Dynamic Approach, Technical Report SR-76, Haskins Laboratory, Yale University, New Haven, CT, 1983.
36. J. L. LaGrange, *Mécanique Analytique*, Gauthier—Villars, Paris, 1788.
37. Sir W. Thompson and P. G. Tait, *Treatise on Natural Philosophy*, Cambridge University Press, Cambridge, UK, 1886.
38. R. Pringle, Jr., "On the stability of a body with connected moving parts," *AIAA* **4**(8), 1395–1404 (August 1966).
39. M. Takegaki and S. Arimoto, "A new feedback method for dynamic control of manipulators," *ASME J. Dynam. Syst. Meas. Contr.* **102**, 119–125 (1981).
40. A. J. Van Der Schaft, Stabilization of Hamiltonian Systems, Memo 470, Technische Hogeschool Twente, Twente, Netherlands, January 1985.
41. D. E. Koditschek, Natural Motion for Robot Arms, *IEEE Proceedings of the Twenty-Third Conference on Decision and Control*, Las Vegas, December, 1984, pp. 733–735.
42. D. E. Koditschek, Natural Control of Robot Arms, Technical Report 8409, Center for Systems Science, Yale University, 1984 (revised March 1985).
43. D. E. Koditschek, Adaptive Strategies for the Control of Natural Motion, *IEEE Proceedings of the Twenty-Fourth Conference on Decision and Control*, Fort Lauderdale, December 1985, pp. 1–4.
44. D. E. Koditschek, "Automatic Planning and Control of Robot Natural Motion via Feedback," in K. S. Narendra (ed.), *Adaptive and Learning Systems: Theory and Application*, Plenum, New York, 1986.
45. R. Brockett, Feedback Invariants for Nonlinear Systems, *Proceedings of the IFAC Congress*, Helsinki, 1978.
46. A. Krener, "On the equivalence of control systems and the linearization of nonlinear systems," *SIAM J. Contr. Optim.* **11**, 670–676 (1973).
47. L. R. Hunt, R. Su, and G. Meyer, Design for Multi-input Nonlinear Systems, in R. W. Brockett, R. S. Millman, and H. J. Sussman (eds.), *Differential Geometric Control Theory*, Birkhauser, Boston, MA, pp. 268–297, 1983.
48. A. Isidori, *Nonlinear Control Systems: An Introduction*, Volume 72 of *Lecture Notes in Control and Information Science*, Springer-Verlag, New York, 1985.
49. J. L. Casti, *Nonlinear System Theory*, Academic, New York, 1985.
50. J. M. Hollerbach, A Recursive Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation and Complexity, in Brady et al. (eds.), *Robot Motion*, MIT Press, pp. 73–87, 1982.
51. D. E. Orin and W. W. Schrader, "Efficient computation of the

- Jacobian for robot manipulators," *Int. J. Robot. Res.* 3(4), 66–75 (Winter 1984).
52. R. H. Lathrop, "Parallelism in manipulator dynamics," *Int. J. Robot. Res.* 4(2), 80–102 (Summer 1985).
  53. C. H. An, C. G. Atkeson, J. D. Griffiths, and J. M. Hollerbach, Experimental Evaluation of Feedforward and Computed Torque Control, *Sixth CISM-IFTOMM Symposium on Theory and Practice of Robots and Manipulators*, September 1986.
  54. E. Freund, "Fast nonlinear control with arbitrary pole placement for industrial robots and manipulators," *Int. J. Robot. Res.* 1(1), 65–78 (1983).
  55. J. Y. S. Luh, M. W. Walker, and R. P. Paul, "Resolved acceleration control of mechanical manipulators," *IEEE Trans. Autom. Contr.* AC-25, 468–474 (1980).
  56. T. J. Tarn, A. K. Bejczy, A. Isidori, and Y. Chen, Nonlinear Feedback in Robot Arm Control, *Proceedings of the Twenty-Third IEEE Conference on Decision and Control*, Las Vegas, NV, December 1984, pp. 736–751.
  57. V. I. Utkin, "Variable structure systems with sliding mode: A survey," *IEEE Trans. Autom. Contr.* AC-22(4), 212–222 (April 1977).
  58. K.-K. D. Young, "Controller design for a manipulator using theory of variable structure systems," *IEEE Trans. Syst. Man, Cybernet.* SMC-8(2), 101–109 (February 1978).
  59. J.-J. E. Slotine, "The robust control of robot manipulators," *Int. J. Robot. Res.* 4(2), 49–64 (Summer 1985).
  60. J.-J. E. Slotine and S. S. Sastry, "Tracking control of nonlinear systems using sliding surfaces, with applications to robot manipulators," *Int. J. Contr.* 38(2), 465–492 (1983).
  61. D. E. Koditschek, Robot Kinematics and Coordinate Transformations, *IEEE Proceedings of the Twenty-Fourth Conference on Decision and Control*, Fort Lauderdale, FL, December 1985, pp. 1405–1409.
  62. M. Spivak, *A Comprehensive Introduction to Differential Geometry*, Publish or Perish, Berkeley, CA, 1979.
  63. D. F. Golla, S. C. Garg, and P. C. Hughes, Linear State Feedback Control of Manipulators, in M. Brady, J. Hollerbach, T. Johnson, T. Lozano-Perez, and M. Mason (eds.), *Robot Motion: Planning and Control*, MIT Press, Cambridge, MA, pp. 169–182, 1982.
  64. M. Vukobratovic, General Software for Computer Aided Synthesis of Control for Manipulator Robots, in M. Brady and R. Paul (eds.), *Robotics Research. The First International Symposium*, MIT Press, Cambridge, MA, pp. 767–781, 1984.
  65. E. Gilbert and D. W. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles," *IEEE J. Robot. Autom.* RA-1(1), 21–30 (March 1985).
  66. A. J. Koivo and T. H. Guo, "Adaptive linear controller for robotic manipulators," *IEEE Trans. Autom. Contr.* AC-28(2), 161–171 (February 1983).
  67. C. P. Neuman and H. W. Stone, Mrac Control of Robotic Manipulators, in K. S. Narendra (ed.), *Proceedings of the Third Yale Workshop on Applications of Adaptive Systems Theory*, Center for Systems Science, Yale University, New Haven, CT, pp. 203–210. June 1983.
  68. M. Tomizuka, R. Horowitz, and C. L. Teo, Model Reference Adaptive Controller and Robust Controller for Positioning of Varying Inertia, *Conference on Applied Motion Control*, Minneapolis MN, June 1985, pp. 191–196.
  69. S. Arimoto, Mathematical Theory of Learning with Applications to Robot Control, in K. S. Narendra (ed.), *Adaptive and Learning Systems: Theory and Applications*, Plenum, New York, 1986.
  70. H. Asada, T. Kanade, and I. Takeyama, "Control of a direct drive arm," *ASME J. Dynam. Syst. Measur. Contr.* 105(3), 136–142 (1983).
  71. J. J. Craig, P. Hsu, and S. Sastry, Adaptive Control of Mechanical

Manipulators, Memorandum M86/3, Electronics Research Laboratory, University of California, Berkeley, Berkeley, CA, Jan 1986.

72. R. Bellman, *Introduction to Matrix Analysis*, McGraw Hill, New York, 1965.

D. E. KODITSCHKEK  
Yale University

This work is supported in part by the National Science Foundation under grant no. DMC-8505160.

## ROBOTICS

The objective of this entry is to survey the state of the art of intelligent robots. By way of introduction, the terms "robot" and "artificial intelligence" are defined, intelligent robots are classified according to their level of intelligence, social and technoeconomic incentives for the development of intelligent robots are discussed, and the entry touches on the socioeconomic impacts of this development. Past accomplishments and present issues in major robotics research areas are covered. A list of relevant books, conference proceedings, and journals is given in the General References.

### Robot Characteristics

**Robot Capability, Components, and Intelligence.** Several definitions for the term "robot" have been proposed in the past (1). None of these definitions are adequate because they exclude robot intelligence of any kind. Hence, the following definition is proposed (2):

*A robot is a general-purpose machine system that, like a human, can perform a variety of different tasks under conditions that may not be known a priori.*

Being a general-purpose machine system, the terms "robot" and "robot system" are regarded as synonymous. A robot system may include any of the following major functional components:

effectors—"arms," "hands," "legs," "feet";

sensors—contact, noncontact;

computers—top-level controller, lower level controllers (including communication channels); and

auxiliary equipment—tools, jigs, fixtures, tables, pallets, conveyors, part feeders, etc.

A robot (or robot system) is controlled by a single top-level computer (or controller). A group of such systems, which may or may not interact, are regarded as separate robots if they are not controlled by the same top-level computer; adding a single computer above them will merge these systems into a single robot.

Although a robot performs some human tasks and there is a similarity between the functional components of a robot and those of a human (or a human team), a robot is not required to act or look like a human. It should, however, be able to perform tasks that require flexibility and artificial intelligence. "Flexibility" means the ability to perform a class of different tasks; "artificial intelligence" means the ability of a machine system to perceive conditions that may not have been known a

priori, decide what actions should then be performed, and plan these actions accordingly. Some of the potential robot tasks can be performed by humans; others cannot (e.g., in a high radiation environment).

**Robot Classification.** Like human intelligence, robot intelligence is variable. This observation is compatible with the Japanese classification of industrial robots into five categories (3):

- a slave manipulator teleoperated by a human master;
- a limited-sequence manipulator (further classified into "hard-to-adjust" and "easy-to-adjust" categories);
- a teach-replay robot;
- a computer-controlled robot; and
- an intelligent robot.

**Learning from Biological Systems.** Current robot capabilities to act, sense, and think are in many respects inferior to those of animals in general and humans in particular. By studying biological systems, one may discover principles that can be used, perhaps by analogy, to improve the functional components of a robot as well as their cooperation. Using such a bionic approach may lead to improvements in robot effectors (e.g., the automatic feedback control of cooperative flexible arms, fingers, and legs or the dexterity of multifinger hands) (see Manipulators), sensors (e.g., integration of several sensors in parallel) (see Multisensor integration), and computer processing (e.g., representation of knowledge (see Representation, knowledge) and reasoning (qv) about it). These improved capabilities will advance sensor-guided manipulation (e.g., picking one of jumbled objects in a bin), perception (e.g., recognizing, locating, and inspecting objects in cluttered environments or outdoor scenes), and other activities in which these components are integrated.

**Incentives for Intelligent Robot Development.** The important incentives for the developments of intelligent robots are social and technoeconomic.

**Social Incentives.** The most important incentive for developing robots should be social—replacing humans who perform undesired jobs by machines. Japan, e.g., is planning to embark on a large-scale program for the development of robots operating in hazardous environments (4). The ranking of robot development should thus be ordered according to job undesirability, i.e., jobs that are

- lethal, e.g., in high radiation environment;
- harmful, e.g., paint spraying, handling toxic chemicals;
- hazardous, e.g., combat, fire fighting;
- strenuous, e.g., lifting heavy loads, visual inspection;
- noisy, e.g., forging, riveting; or
- dull, e.g., sorting, assembling.

**Technoeconomic Incentives.** The second most important incentive for robot development is technoeconomic—reducing the cost of manufacturing products and improving their quality.

**Current Limitations.** In spite of the strong social and economic incentives mentioned above, only a very small fraction of the entire human workforce in the world has been replaced by industrial robots. Furthermore, Engleberger (5) estimates

that the growth rate of the total number of industrial robots (excluding teleoperators (qv) and limited-sequence manipulators) will rise from 2000 per year in 1980 to 40,000 per year in 1990; these figures correspond to a yearly replacement of about 0.003–0.06% of the total blue-collar workforce in the industrialized countries. Such a low rate of growth of robot population has resulted primarily from the following limitations of today's industrial robots (2).

**Insufficient Material-Handling Flexibility.** Workpieces and other objects can be handled only if they are indexed within tolerances that match the accuracy of the robot manipulator. Such restriction limits the flexibility of manufacturing, especially in batch production of a mix of products.

**Open-Loop Control.** Jobs that require closed-loop feedback control to correct local errors cannot be performed. For example, today's arc-welding robots cannot track a joint of randomly variable shape and gap in one pass and adjust the torch movement and welding parameters accordingly; this limitation excludes these robots from a large market.

**Inability to Detect and Correct Errors.** Detection of unexpected errors and the recovery from them cannot be done; a robot system cannot verify that all the robot actions have been executed as planned. The resulting penalty may be costly. For example, the cost of debugging and repairing a final assembly may be several orders of magnitude higher than the cost of correcting that error in process, whereas subassemblies are easily accessible.

**Restricted Mobility.** The locomotion of today's robotic carts is restricted to fixed guidance (e.g., by buried cables or painted lines); these carts cannot navigate freely, avoid obstacles, or find their targets in an unstructured environment. Such a restriction limits the flexibility of material handling in batch production (see Autonomous vehicles; Robots, mobile).

**Future Capabilities.** The best way to overcome the limitations of today's "muscle-only" robots is to provide them with intelligence, i.e., adaptive sensing and thinking capabilities. Such intelligent robots will be able to compete more effectively with not only blue-collar workers but also white-collar workers. Most industrial companies have not yet agreed with this observation but they will, eventually, when the threat of worldwide market competition becomes unbearable.

**Socioeconomic Problems.** Development of intelligent robots may raise many problems, the major one of which is unemployment (77). Obstructing development of intelligent robots by the labor unions will only worsen the unemployment problem because other countries, especially Japan, will proceed with such development and, as a result, foreign competition will become stronger. This complex problem will probably be mitigated by three factors (6):

- new related jobs—an increased demand for skills related to intelligent robots directly (e.g., engineering, computer programming, and manufacturing) and indirectly (e.g., professional training, marketing, shipping, and servicing);
- new unrelated jobs—a shift to other jobs, especially in the service industry (thus raising the standard of living); and
- fewer working hours—reducing the working hours per week with no reduction in the standard of living.

Whether these factors will be able to solve the unemployment problem remains to be seen. In the meantime, the current rate of intelligent-robot development is low, amounting to robot evolution rather than robot revolution. Such evolution will enable society to adjust gradually, without adverse repercussions, to the advent of the intelligent robot.

**Technical Approach.** The technical approach to intelligent-robot development should be based on the application of AI techniques to robotics under four engineering constraints:

high reliability—the robot must be robust; if it fails, it should be able to detect the error and recover from it or call for help;

high speed—the robot should be able to perform its functions as fast as necessary;

programmability—the robot should be flexible (able to perform a class of different functions for a variety of tasks), easily trainable (for new tasks or modification of old ones), and intelligent (able to perceive problems and solve them); and

low cost—the cost of the robot should be low enough to justify its application.

Clearly, these constraints may conflict with each other. For example, increasing the robot speed or lowering its cost may also lower its reliability. A trade-off, therefore, must be engineered for different applications according to the significance of each constraint.

**Research-and-Development Topics.** As shown above, a robot system may be divided into effectors, sensors, computers, and auxiliary equipment. Robotics R&D topics associated with these major functional components include manipulation (of arms), end effectors, and mobility; sensing (in general), non-contact sensing, and contact sensing; adaptive control (which utilizes sensors to monitor and guide effector actions); robot programming and manufacturing process planning (which generate task-specific computer programs that are executed by the top-level and lower level controllers). Past achievements and research issues related to each of these topics are described briefly in the following sections.

## Manipulation

Robot manipulation entails the kinematics, motion trajectories, dynamics, and control of a robot arm (see Manipulators).

**Kinematics.** The location (position and orientation) of a robot wrist in a frame attached to the base of the robot arm is described in two ways:

joint coordinates—the angles of the rotary joints and the lengths of the sliding joints of the arm; and

world coordinates—the Cartesian coordinates of the wrist position and the direction cosines defining the wrist orientation.

Joint coordinates must be used to command the robot arm to arrive at a given wrist location. On the other hand, humans prefer to describe the wrist location in terms of world coordi-

nates. Hence, means are provided for transforming from one set of coordinates to another.

**Joint-to-World Coordinate Transformation.** A frame of Cartesian coordinates  $(x, y, z)$  is attached to every arm joint according to a set of rules proposed by Denavit and Hartenberg (7). The homogeneous coordinates  $(x, y, z, 1)$  of a given point in each joint frame are converted to those of a neighboring one by means of a transform—a  $4 \times 4$  homogeneous coordinate-transformation matrix. Multiplying the transforms of all the arm joints results in the arm-to-wrist transform whose elements, expressed in terms of the joint coordinates, describe the position and orientation (direction cosines) of the arm wrist (8,9).

**World-to-Joint Coordinate Transformation.** Given the position and orientation of the wrist of an arm, solving for the corresponding joint coordinates is less systematic and more difficult than vice versa. Each arm has a unique solution in which joint coordinates are computed sequentially in a fixed order (8).

## Motion Trajectories

**Work-Station Transforms.** Denoting the homogeneous coordinates  $(x, y, z, 1)$  of a point  $P$  in a frame  $F$  by  $P(F)$ , then  $P(F_1) = [F_1/F_2] * P(F_2)$ , where  $[F_1/F_2]$  is the  $4 \times 4$  transform from frame  $F_1$  to frame  $F_2$  (see Fig. 1a). Note that  $P(F_2) = [F_2/F_1] * P(F_1)$ , where  $[F_2/F_1]$  is the inverse of  $[F_1/F_2]$ .

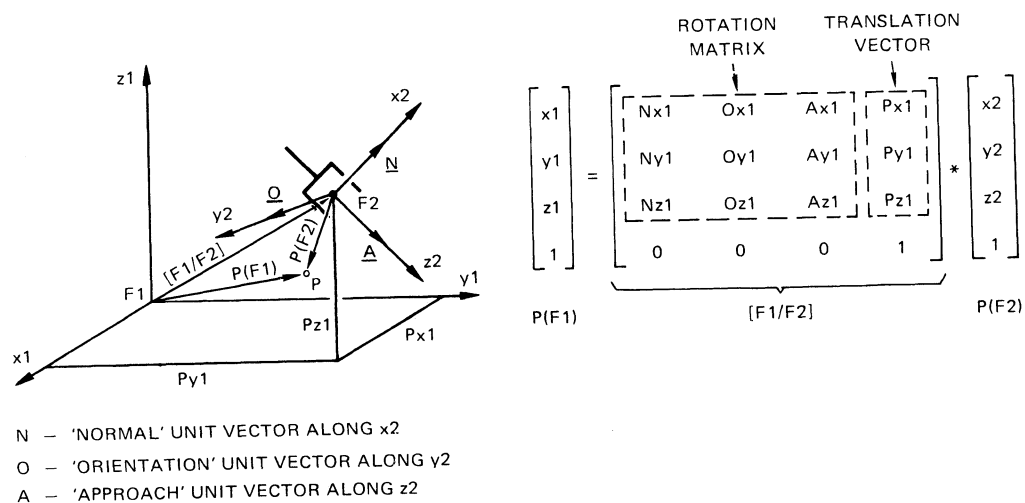
Consider a robot arm with an end-effector mounted on its wrist moving in a work station, and let  $A$ ,  $W$ ,  $E$ , and  $C$  denote frames attached to the arm, the wrist, the end-effector, and the current (instantaneous) action target of the robot, respectively. Given transforms  $[A/C]$ ,  $[C/E]$ , and  $[W/E]$ , the unknown transform  $[A/W]$  is computed from the relation  $[A/W] = [A/C] * [C/E] * [E/W]$ , where  $[E/W]$  is the inverse of  $[W/E]$  (see Fig. 1b). Knowing  $[A/W]$ , the corresponding joint coordinates are then computed using the world-to-joint arm solution.

**Smooth Path.** Paul (10) developed a technique for moving the robot end effector along a path consisting of straight segments and smooth transitions between them (see also Refs. 9, 11, and 78). A straight segment is obtained by interpolating world and joint coordinates between its end locations while maintaining a constant linear velocity. A smooth transition between two straight segments is obtained by bypassing their intersection point along a parabolic curve tangent to both segments. Each pair of straight segments may be defined in either a single frame or in two frames, each of which may be moving with a constant velocity relative to the arm frame. For example, a robot spot-welding gun may move with velocity  $V_1$  along a straight line in the arm frame toward a moving conveyor, veer and change its velocity smoothly, and continue to move with velocity  $V_2$  along a second straight line in the conveyor frame toward a welding spot. Training for a task to be performed on a moving line is done while the workpiece is stationary.

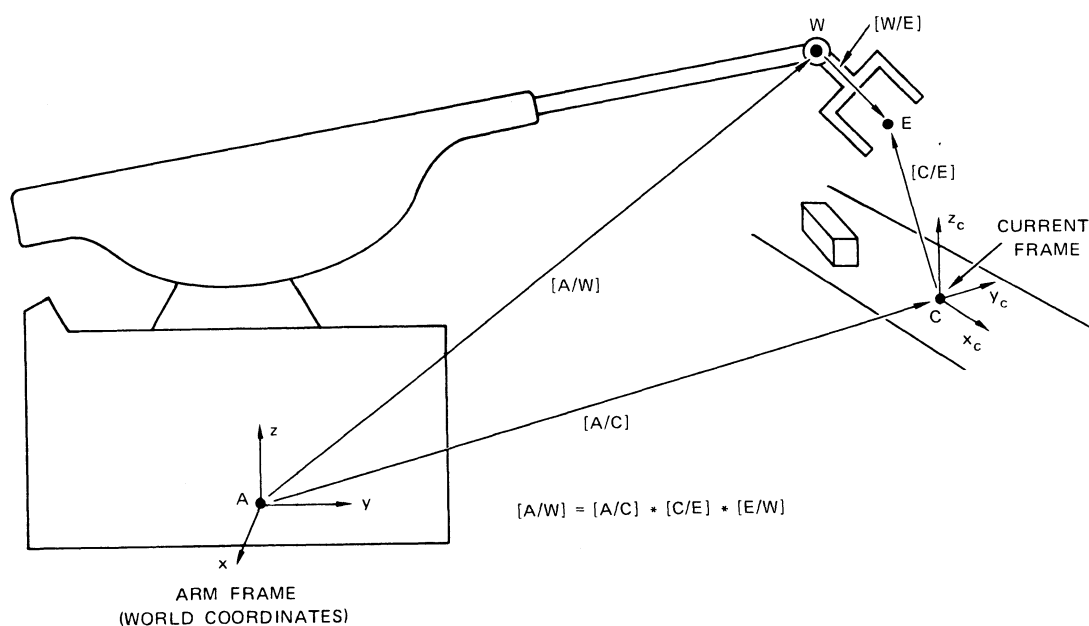
**Dynamics.** The dynamics of a manipulator relate the applied forces/torques to the joint motion (positions, velocities, and accelerations as functions of time). Two dynamics problems are distinguished (12):

the "forward problem"—given the applied forces and torques, solve for the resulting joint motions; and

the "inverse problem"—given the joint motions, solve for the required forces/torques.



(a) TRANSFORMATION MATRIX



(b) MATRIX DESCRIPTION OF ARM SYSTEM

Figure 1. Arm-transformation matrices.

The inverse problem is more important for real-time arm control and, fortunately, easier than the forward problem.

Different numerical methods have been proposed for solving the equations of motion of an  $n$ -joint manipulator. These methods are assessed on the basis of their computational complexity (the number of additions and multiplications required). The methods are based on either the Lagrangian formulation (9,13) or the Newton-Euler formulation (14). The Lagrangian formulation is less efficient, but Silver (15) showed that the two formulations are essentially equivalent and that the complexity of computation depends on its structure and the representation of the rotational dynamics. Kane and Levinson (16) proposed another approach that uses explicit dynamic equations without the unnecessary computations entailed in either of the above formulations and, hence, should be more efficient.

**Control.** The motion of a manipulator in a free space is usually controlled by means of a position servo in each joint. If the manipulator is required to move while exerting a specified force/torque on an object, appropriate force/torque servo and position servo must be executed simultaneously (9). Such a hybrid control can be achieved by replacing the position servos of selected joints by force/torque servos so that the manipulator is free to move in the specified direction while the prescribed forces/torques are applied to the object-surface normals. For example, grinding a horizontal surface requires simultaneous control of the position of the grinding wheel and the vertical force it exerts. As another example, inserting a peg into a hole requires motion along two directions normal to it. Stopping the motion of a manipulator can be controlled by specifying a given force/torque condition.



## End-Effectors

An end-effector is a functional device attached to the wrist of a robot arm. Four types of end-effector are distinguished: hand, tool, hand/tool holder, and micromanipulator. Each of these should be small, light, fast, accurate, multifunctional, and inexpensive.

**Hand.** A hand, whose major function is to grasp objects, includes a number (e.g., two or three) of fingers attached to its "palm." Each finger should have humanlike structural dexterity and rigidity ("bone"), object-grasping compliance ("flesh"), surface tactile sensors ("skin"), and proximity sensors for collision avoidance (no human equivalence). The hand may be equipped with a wrist force sensor, a visual sensor ("eye"), a range sensor, or any other sensor. To minimize the number of wires between the hand and the robot controller, local signals and hand functions should be processed by microprocessors mounted on the hand itself (resulting in a "smart hand").

Three-finger hands have been built at the Electrotechnical Laboratory, Tsukuba, Japan (17), and at Stanford University (18). The Salisbury hand is shown in Figure 2. A kinematic analysis of the latter hand yields a large number (373) of different ways the hand can grasp an object. Hand control is very complex; it is equivalent to the control of three cooperating three-joint arms with force sensors.

**Tool.** A tool may be a spot-welding gun, an arc-welding torch, a wrench, or any other device that performs a certain task. If a sensor is mounted on or near the tool, it must be ensured that neither one will prevent the other from access to its target. For example, using visual sensing to guide a robot to arc weld workpieces with corners may require that the sensor and the welding torch be free to move relative to each other.

**Hand/Tool Holder.** A hand/tool holder is a device mounted between the arm wrist and a hand or a tool for one of two purposes:

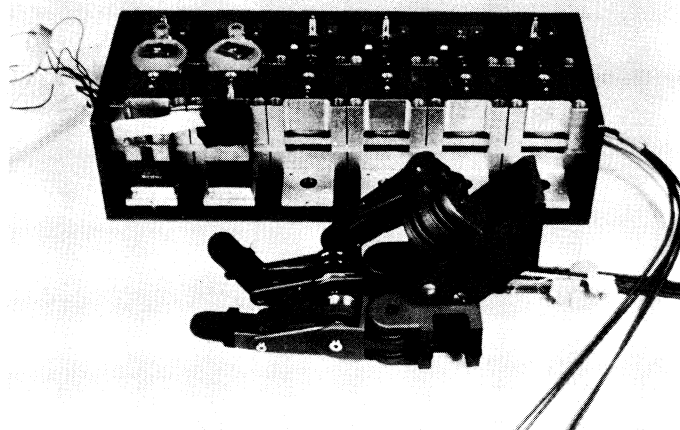
- quick hand/tool changing and mounting for different tasks, which may be achieved by a standard latch/unlatch device or a hand; and

- local accommodation, which may be implemented passively by a remote center compliance (RCC) device, developed at Draper Laboratories (19), or actively by a force/torque sensor. For example, in Figure 3 plastic parts are assembled by two two-finger PUMA (Unimation, Inc.) hands, one mounted on an RCC (made by Lord Corporation) and guided locally by its "eye" and the other mounted on an xyz force/torque sensor.

**Micromanipulator.** The function of a micromanipulator is to correct a locational error that is measured by a robot sensor and is smaller than the spatial resolution of the arm wrist. Having a much smaller inertia, the micromanipulator is also much faster than the arm. A micromanipulator may be an xyz device or a multifinger hand.

## Mobility

Most industrial robots today are anchored to fixed locations; a few have a limited mobility on tracks mounted on the factory



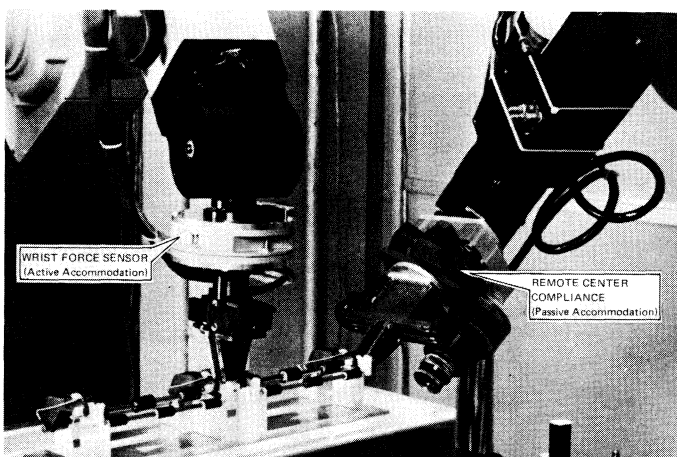
**Figure 2.** Three-finger hand. (Source: Courtesy of Ken Salisbury, MIT.)

floor or on a gantry. There are also mobile carts that transport workpieces, but these carts can move only in a structured environment, e.g., by following buried cables or painted lines. Robot mobility, however, is also needed for a wide variety of robot functions in unstructured environments, such as mining, military operations, and aid to the handicapped. Some robot-mobility issues follow.

**Surfaces and Locomotion.** The mechanism for the robot locomotion depends strongly on the type of surface the robot must be able to move on. Indoor surfaces include floors, ramps, stairs, and cluttered environments. Outdoor surfaces include roads, smooth ground (flat and slanted), terrain with holes and ditches, and terrain with large obstacles.

Robot locomotion is realized with wheels, tracks, and legs. Wheels perform well if the terrain is not rough and the traction is sufficient. Tracks perform well if the terrain slope is not too high or no major obstacles are encountered. In a recent development at Hitachi (20) the loop of each track forms a triangle whose shape is adjusted according to the terrain, thus enabling the vehicle to pass over different obstacles and climb up and down stairways.

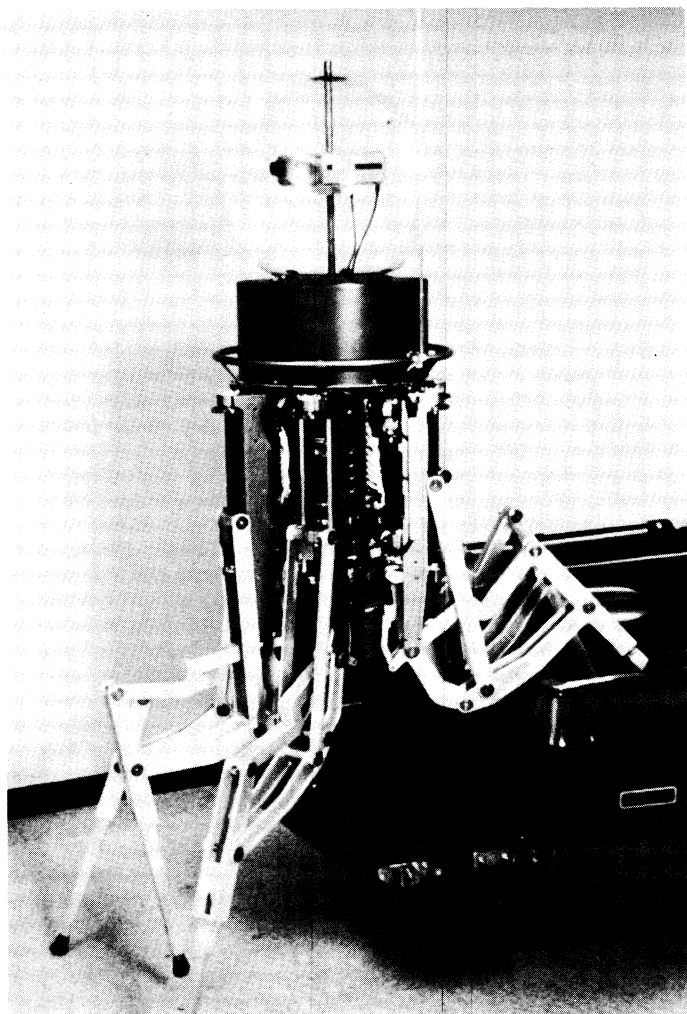
Legged vehicles have been developed for robot mobility in rough terrain, where wheels and tracks are useless. The major



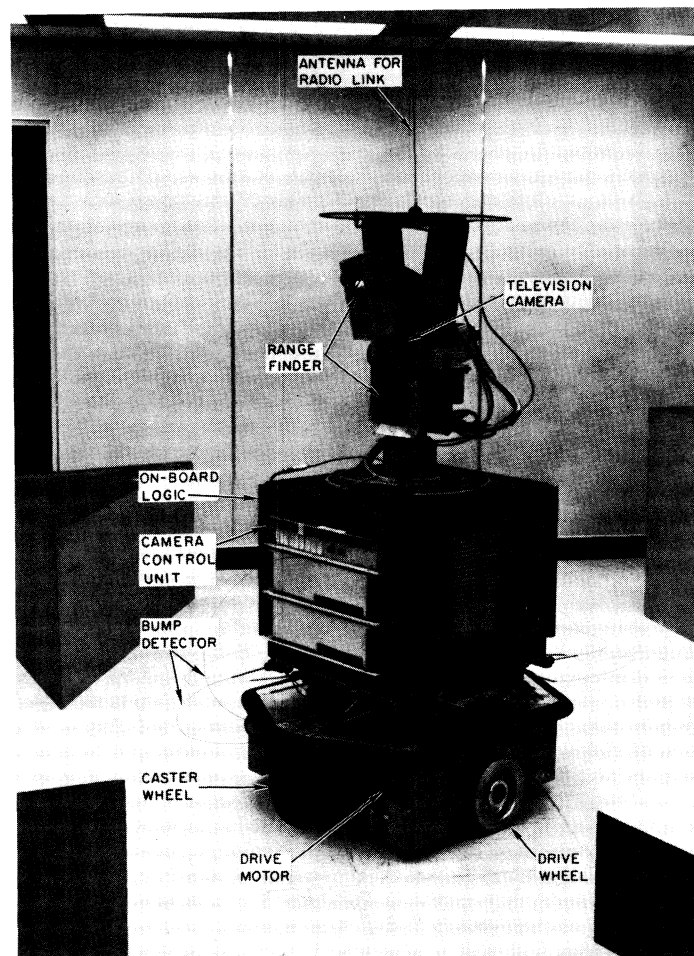
**Figure 3.** PUMA hands with passive and active accommodation devices.

issues are stability, gait, strength, speed, and control. Static stability is achieved if the vertical projection of the center of gravity is within the polygon formed by the vehicle's feet on the ground. A six-legged vehicle with at least three legs always on the ground is inherently stable. Six-legged vehicles have been built at Ohio State University (21), Carnegie-Mellon University (22), and Odetics (23). The Odetics vehicle (see Fig. 4) is characterized by high strength-to-weight ratio and agility.

**Control.** A major research issue in robot mobility is autonomous control, which includes motor control, sensing, navigation, communication, obstacle avoidance, and task performance. SRI's Shakey (qv) (24) was developed in the 1960s as an intelligent mobile robot with these properties. Shakey (see Fig. 5) had autonomous wheel-drive control and visual, range, and binary tactile sensors; navigated through laboratory "rooms"; communicated with its "brain" (a DEC PDP-10 computer) via a radio link; avoided obstacles; and pushed boxes according to the plan of a task it was assigned to do. Since then other similar robots have been developed in the United States (e.g., at Carnegie-Mellon University and MIT), France (e.g., LAAS in Toulouse), and Japan (e.g., Mechanical Engineering Laboratory in Tsukuba). These robots use wheels to move on laboratory floors and shaft encoders to sense their 2-D posi-



**Figure 4.** Six-legged walking vehicle. (Source: Courtesy of Odetics, Inc., Anaheim, California.)



**Figure 5.** SRI's Shakey, a mobile robot.

tions. To correct for random locational errors, some of them use infrared beacons, a directable laser range finder, and visual perception. Sonar sensors (which are inexpensive) are commonly used to avoid obstacles, but they are not adequate for navigation because of their poor resolution, short range, and vulnerability to specular reflection.

The primary research issues in planning for and execution of robot mobility along long-range paths are representation and mapping of a 3-D "world," noncontact sensing, outdoor visual perception, navigation, and communication, and those along short-range paths are noncontact and contact sensing, obstacle avoidance, foothold location (to avoid holes, ditches, and the like), and recovery from accidental falls (see Autonomous vehicles; Robots, mobile).

### Robot Sensing

**Sensing Sequence.** Robot sensing is defined as perception—translation of relevant characteristic or relational object properties into the information required to control the robot in performing a given robot function (2). The object properties may be geometric, mechanical, optical, acoustic, material, electric, magnetic, chemical, and the like. Robot functions may be passive (e.g., RECOGNIZE, LOCATE, and INSPECT) or active (e.g., GRASP, TRANSPORT, and WELD). Each of these functions may be expanded by lower level robot functions (e.g., RECOGNIZE = (TAKE-PICTURE, FIND-EDGE, EXTRACT-FEATURES, INTERPRET-FEATURES)), or be used to define

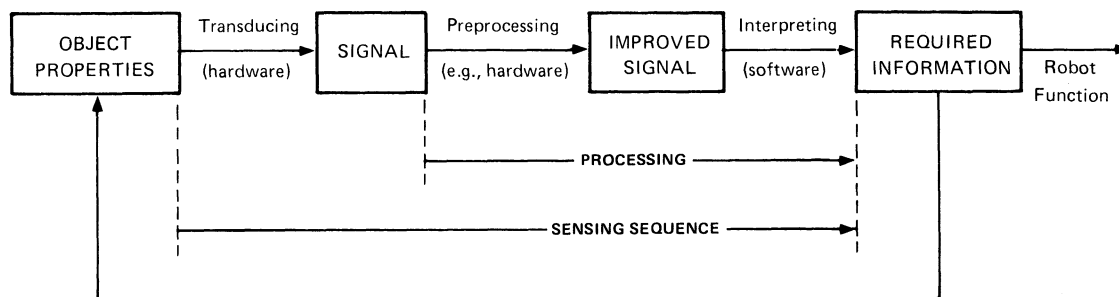


Figure 6. Robot-sensing diagram.

higher level ones (e.g., using the functions FIND, ACQUIRE, HOLD, MOVE, ALIGN, INSERT, and VERIFY to define ASSEMBLE). As shown in Figure 6, a robot sensing sequence is performed in the following steps.

1. *Transducing*—converting (in hardware) the relevant object properties into a signal.
2. *Processing*—transforming the signal into the required information, usually in two substeps: *preprocessing*, improving the signal (usually in hardware), e.g., filtering out noise, and *interpreting*, analyzing the improved signal and extracting the required information (usually in software).

The above steps probably cannot be implemented by a general-purpose system. Instead, each step or substep should be performed by effective hardware/software schemes, regarded as tools, that depend on the environmental conditions and the specified robot functions. To be able to carry out a wide variety of sensing tasks, the sensing system will consist of sets of tools, or tool boxes, and a knowledge-based "supervisor" that can select the best tools for each given task.

**Sensing Strategy.** If the extracted information is not sufficient, the sensing sequence is modified and repeated in order to obtain complementary information. Three cases are distinguished as follows.

1. *Supplementary images*—imaging additional surfaces, which are hidden from a fixed sensor in the previous sensing sequence(s), by means of a sensor mounted on the robot end effector or multiple sensors at different viewpoints.
2. *Sensing efficiency*—achieving efficient sensing by first using coarse resolution and then fine resolution, such as: recognizing an object, then locating it precisely; and recognizing and locating an object approximately, then inspecting some of its "windows," where distinctive features or defects may be found, with fine resolution.
3. *Multisensing*—utilizing different sensors to supplement a sensor output, such as recognizing and locating an object with vision and then locating it precisely and verifying its grasp with tactile sensing, i.e., verifying the sensing of one sensor by another.

**Sensor Signals.** A given object property may be measured by different sensor signals, such as light intensity, range, acoustic, tactile, force, and temperature (see Table 1). A point signal is distinguished from an array (1-D or 2-D) of point signals. As shown in Table 2, each of these signals may be generated by different sensor transducers; e.g., a point light intensity may

be transduced by a photocell, a photomultiplier, a 1-D array, or a 2-D array.

**Research Issues.** Robot sensing could be advanced by developing

- sensor transducers that have a higher resolution, higher speed, smaller size, and lower cost;
- faster hardware/software processors that can process a larger amount of sensor signals and extract more information; and
- sensor modeling and planning, including sensor selection, for a given task and off-line signal prediction.

### Noncontact Sensing

Noncontact sensing is based on a signal generated by a transducer that is not in physical contact with the object it senses. We classify noncontact sensing according to the type of signal, i.e., light intensity (or, briefly, intensity), range, acoustic, temperature, chemical, etc. Noncontact sensing for robot applications has so far been based primarily on intensity and range signals; future robot applications should also utilize the other types of signals.

The output of the signal-transduction step is an image if it consists of a 2-D array of sensory data values. We thus distinguish between an intensity image, which consists of  $N \times M$  picture elements, or pixels, and a range image, which consists of  $N \times M$  range elements, or rangels, where  $N$  and  $M$  are resolution integers. An intensity image provides information about the reflectance of object surfaces in the scene, but it may be ambiguous geometrically because of the loss of 1-D information in the process of transforming a 3-D world into a 2-D gray-level image. A range image, on the other hand, provides 3-D geometric information directly but no reflectance information. Intensity and range images are thus complementary and should, therefore, be in exact registration to simplify the analysis of their integrated information.

Intensity transducers include a photocell, a photomultiplier, a 1-D array camera, and a 2-D array camera. Intensity transducers require the following improvements (25):

- chips with higher precision, improved quality, color discrimination, higher resolution (e.g.,  $1024 \times 1024$  or even  $2048 \times 2048$  pixels);
- lenses with lower distortion and better focus in the infrared region; and
- fast, computer-controlled adaptive lens opening and focusing as well as intensity thresholding.

**Table 1. Measurement of Object Properties: Possible Signals for Each Object Property**

Object Property	Signal					
	Intensity (Point/Array)	Range (Point/Array)	Acoustic (Point/Array)	Tactile (Point/Array)	Force (Point)	Temperature (Point/Array)
Geometric						
Centroid	×	×		×		
Edge, Corner	×	×	×	×		
Surface	×	×	×	×		
Volume		×	×			
Width	×	×	×	×		
Texture	×	×	×	×		
Shape	×	×	×	×		
Proximity		×	×	×		
Mechanical						
Weight					×	
Force/Torque					×	
Pressure				×	×	
Optical						
Reflectance	×	×				
Color	×	×				
Acoustic						
Reflectance			×	×		
Material						
Hardness				×	×	×
Temperature						×

Indirect measurement of range or surface orientation may be inferred from monocular 2-D images under certain conditions (26–28), but this subject is beyond the scope of this chapter. The following section focuses on direct range measurement and its current problems.

**Direct Range Measurement.** Two basically different techniques can be used to measure range directly—triangulation and time of flight.

**Triangulation Techniques.** Triangulation is based on elementary geometry (see Fig. 7a): given the baseline of a triangle, i.e., the distance between two of its vertices, and the angles at these vertices, the range from one of the vertices to the

third one is computed as the corresponding triangle side. Triangulation techniques are subdivided into two schemes: stereo, using ambient light and two cameras (a passive scheme), and structured light, using a projector of controlled light and camera (an active scheme). The plane in which the triangle lies is called the epipolar plane, and its line of intersection with a camera image plane is called the epipolar line.

The main drawbacks of any triangulation technique are missing data for points in the scene that are not “seen” from both vertices of the triangle. This problem can be partially solved in two ways:

decreasing the baseline (this remedy, however, will increase the measurement errors) and

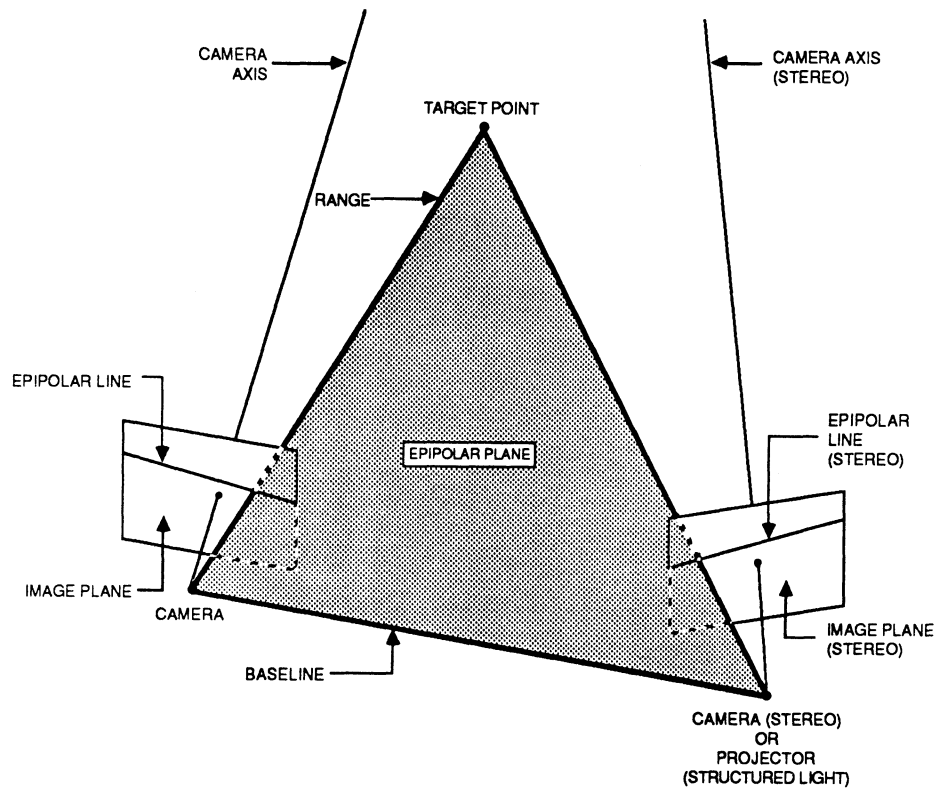
using multiple cameras for the stereo scheme or multiple projectors and/or cameras for the structured light scheme (this provision will also reduce the measurement errors and mitigate the problem of occlusion, including self-occlusion, in machine vision, but it will increase the cost, complexity, and measurement time of the system).

**Stereo.** Relying on passive ambient light, triangulation stereo techniques use an image sensor (in particular, a TV camera) at each of two triangle vertices. A stereo pair of images can be obtained either from two static cameras (at different locations) or from one camera that is moved between two locations.

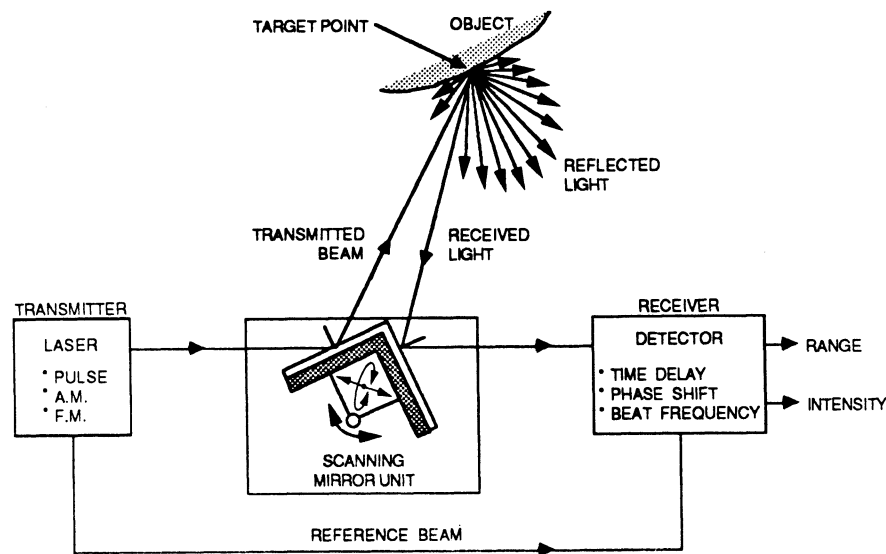
In addition to the missing-data problem, the main issue in stereo vision is the correspondence problem: how to match corresponding points in stereo images reliably and quickly. This problem has no solution if the two images have uniform reflectance. Conversely, the correspondence problem becomes easier as the stereo images include more intensity features, such as edges, especially if they are perpendicular to the epipolar line. These features should be extracted on the basis of microconstraints as well as macroconstraints. For example,

**Table 2. Measurement of Object Properties: Possible Sensor Transducers for Each Signal**

Signal	Transducers
Intensity	
Point	Photocell; photomultiplier; array (1-D; 2-D)
Array	Array or equivalent (lower dimensional array scanning)
Range	
Point	Projector (laser; planar light)/receiver (photomultiplier; array; two arrays); acoustic
Array	Scanning projector (laser; planar light)/receiver (photomultiplier; array); two 2-D arrays or equivalent
Acoustic	
Point	Acoustic transducer
Array	Array of acoustic transducers or equivalent
Tactile	
Point	Microswitch; array of tactile transducers
Array	Array of tactile transducers or equivalent
Force (point)	Force transducer
Temperature	
Point	Thermocouple; infrared transducer
Array	Array of infrared transducers or equivalent



(a) TRIANGULATION RANGE SENSING



(b) TIME-OF-FLIGHT LASER RANGE SENSING

Figure 7. Direct range-sensing schemes. (a) (Source: Ref. 31.) (b) (Source: Courtesy of R. C. Bolles, SRI International.)

local intensity changes imply edge points, but if these points are too isolated to be linked into a continuous edge, they should be disregarded. The effect of the correspondence problem is an increase in the measurement time.

*Structured Light.* One way to dispose of the correspondence problem is to use active light—a scheme in which one of the stereo cameras is replaced by a source of specially controlled illumination, called structured light. The structured light may

be projected serially, by scanning a collimated light beam (usually a laser), or in parallel, either by diverging a laser beam with a cylindrical lens or by using a slit or a slide projector. The structured light may consist of single or multiple light patterns, each of which may be a straight line (a beam), planar, or nonplanar.

In addition to the missing-data problem, the structured light scheme entails two issues:

**Specular Reflection.** Reflection from a mirrorlike surface may result in no range measurement if the reflected light does not reach the camera and false (larger or smaller) measured range values if the reflected light is subsequently reflected by other surfaces before part of it reaches the camera.

**Slow Measurement.** Serial projection of multiple light planes requires too much time for data acquisition. This problem can be mitigated by projecting them in parallel, but this entails determination of the correspondence between each light plane and the image of its intersection with the target. As a trade-off between serial vs. parallel projection, the following time-coded light pattern projection method was proposed (29): each plane among a set of different light planes is turned on or off during each of a sequence of time slots according to a given code, and the resulting images are decoded to determine the correspondence between each plane and its image.

**Time-of-Flight Techniques.** A time-of-flight range sensor (see Fig. 7b) includes a signal transmitter and a signal receiver consisting of a collector of part of the signal reflected by the target and the electronics for measuring the round-trip travel time of the returning signal and its intensity. Two types of signal are practical: ultrasound (such as used by the Polaroid range sensor) and laser light. Ultrasound is much more adversely affected by surface specularities and has a much poorer spatial resolution than laser light; hence, let us consider only laser light.

Time-of-flight laser range sensors use a scanning mirror to direct the transmitted laser beam along pan-and-tilt orientations with equal angular increments in order to obtain a range image consisting of  $N \times M$  rangels. Like with triangulation range sensing, by also measuring the intensity of the reflected light, we obtain an intensity image consisting of  $N \times M$  pixels in complete registration with the range image. On the other hand, the missing-data problem that is inherent in triangulation range sensing is eliminated by mounting the laser transmitter coaxially with the receiver's reflected light collector.

Three schemes can be distinguished for measuring the length of the transmitter-target-receiver optical path in time-of-flight laser range sensing (see Fig. 7b):

1. **Pulse Time Delay.** Using a pulsed laser and measuring the time of flight directly (30). This scheme requires advanced electronics.
2. **AM Phase Shift.** Using an amplitude-modulated laser and measuring the phase shift, which is proportional to the time of flight (31).
3. **FM Beat.** Using "chirps" of laser waves that are frequency modulated as a linear function of time and measuring the beat frequency, which is proportional to the time of flight (32).

Time-of-flight laser range sensors have the following problems:

**Specular Reflection.** Reflection from a mirrorlike surface may result in no range measurement if the reflected light does not reach the receiver and larger measured range values if the reflected light is subsequently reflected by other surfaces before part of it reaches the receiver.

**Slow Measurement.** A long integration time is required to reduce the photon noise (and other types of noise) to an acceptable level, especially if the target is dark. For given values of target reflectance, incidence angle, range, and measurement error, the integration time is inversely proportional to the product of the transmitted laser power and the area of the receiver's collector (31).

**Ambiguity in AM Phase Shift.** If the phase shift  $\phi$  between the transmitted light and the received light in an amplitude-modulated scheme may exceed  $2\pi$ , the (true) range  $r$  is ambiguous:  $r = n\lambda + r(\phi)$ , where  $n = 0, 1, 2, \dots$ ,  $\lambda$  is the wavelength of the modulation frequency, and  $r(\phi)$  is the measured range assuming that  $0 \leq \phi \leq 2\pi$ . Detection of range discontinuities, which are unexpected based on the "world knowledge," may be used to determine the value(s) of  $n$ .

Processing of light-intensity images and range images are discussed separately in the following two sections.

**Intensity Image Processing.** Light-intensity image processing in robotic applications is intended primarily to recognize, locate, and inspect objects from an intensity range. Assuming that the image of an object in a given stable state is invariant to its location (position and orientation), processing of three types of intensity image are distinguished below—a complete and isolated outline, a partial or connected (not isolated) outline, and a gray-level image of an object. Processing the intensity images of 3-D objects in general is a more difficult task.

**Complete and Isolated Outline.** The complete outline of an isolated 2-D object or a 3-D object in a given stable state is the boundary between the top-view image (viewed from infinity) of the object and its background. An object outline can be sensed in two ways:

by detecting the black-to-white and white-to-black transitions in a binary image (silhouette)—the main problems here are obtaining a high black-white contrast when using front illumination (it is relatively easy with back illumination) and adjusting the threshold that converts the image into binary pixels dynamically as the task and lighting conditions vary; and

by detecting the edges between dark and bright regions in a gray-level image—edges may be detected by using several methods, such as the Sobel operator (33) and thinning the resulting "thick edges" or by applying the MIT zero-crossing operator (34), which extracts chains of zero-valued pixels from the convolution of the gray levels with the Laplacian of a bivariate Gaussian distribution function. The main problems here are lack of robustness (obtaining false edge points and missing true ones), linking the edge points into expected outlines, and relatively lengthy computation time.

The SRI vision module (35) implements the outline interpretation (recognition, location, and inspection) by matching the features of each "blob," or connected region, of the measured outline with those of a model (prototype). The vision module utilizes many global features, such as the outline's area, perimeter, area-perimeter-squared ratio, radius vectors (minimum, maximum, and average) from the centroid to the perimeter, first and second moments, and number and area of holes. Two recognition schemes are distinguished:



*Nearest neighbor classifier*—selection of the nearest object prototypes in a multifeature space; and

*Decision tree*—sequentially divides object prototypes into two groups along “tree branches” according to the largest gap between the values of the most distinctive feature until reaching a “leaf.”

The SRI vision module (see Fig. 8) consists of three components: one to four TV cameras, a hardware preprocessor, and vision software stored in a microcomputer. Several companies (e.g., Machine Intelligence Corporation and Automatrix Incorporated) have been manufacturing industrial vision systems based on the SRI vision (qv) module.

**Partial or Connected Outline.** Consider 2-D or 3-D objects that may be partially viewed, overlapping without altering their top views substantially or touching. Object outlines are extracted from binary or gray-level images using the same techniques as for an isolated object. The observable outline of any one of these objects may be partial, complete, or connected with another outline; hence, matching cannot be based on global features.

Object recognition and location based on any of these outlines was achieved at SRI (36) by focusing on local features, such as small holes, convex corners, and concave corners, which are much smaller than the image of the entire object. As a local feature is detected, the shape and location of its closest local features are extracted and compared with those of a model until there is sufficient match to hypothesize the object's identity and location. The hypothesis is then verified by comparing the observable-object outline, excluding the local features matched previously, with the model outline. For example, local features extracted from the silhouette of four overlapping or touching door hinges were analyzed by this match/verify system and the resulting outlines, shown in Figure 9, were as expected. The system is robust, but further research is needed to “explain” (by reasoning about object outlines) why portions of some objects are not visible.

**Gray-Level Image.** Techniques for matching the gray-level image of an object with that of a model are covered in several textbooks (e.g., Refs. 33, 37, 38, and 79). These techniques vary over a wide range of complexity, depending on the tasks. One technique, explained previously, is to detect expected edges

between dark and bright regions and match the global or local features of these edges with those of an object model. Two other matching techniques are as follows:

*Average gray-level matching*—the average gray level of a “windowed” (partial) or the entire image of an indexed object (i.e., placed in a known location) is compared with that of a model to determine if there is an acceptable match—simple and fast, this technique is used in commercial visual inspection systems (e.g., Ref. 39); and

*Statistical gray-level matching*—A gray-level histogram (a function showing, for each gray-level increment, the number of pixels in the image whose gray-levels lie within that increment) of an image of an indexed object is compared with that of a model to detect missing or severely damaged parts, and a gray-level joint distribution of the object and its model are used to detect missing or misplaced parts (40).

**Range Image Processing.** Range image processing is intended primarily to recognize, locate, and inspect 3-D objects. Basically, such processing is implemented by extracting 3-D geometric features and matching them with those of a model. Matching the 3-D features of an isolated object whose image is variable is much more difficult than matching the 2-D features of an isolated object whose image is invariable. Matching the features of 3-D objects that are piled on a tray or jumbled in a bin is even more difficult because each object may have infinite possible orientations and may be partially occluded by other objects. Compared with visual image processing, range image processing has barely “scratched the surface”; much research is still needed in this area.

Since range data describe the surface of a solid object (not its interior), the object should be represented by its surface (not its volume). Three useful representations of 3-D surfaces are considered below: faces, generalized cylinder surfaces, and volumetric.

**Faces.** Object faces are usually represented mathematically by a number of unbounded planar or curved surfaces (e.g.,



Figure 8. The SRI vision module.

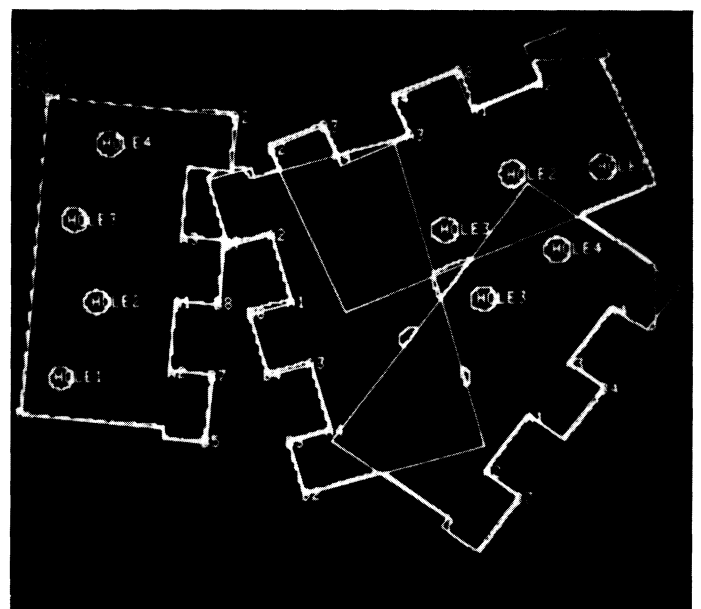


Figure 9. Using local features to locate overlapping or touching door hinges. (Source: Courtesy of R. C. Bolles, SRI International.)

cylindrical, conic, or spherical) that confine the object. These surfaces may intersect along edges which, in turn, may intersect at vertices. Geometric modeling in most computer-aided-design (qv) (CAD) systems is based on this representation.

Geometric features (surfaces, edges, and vertices) "seen" by a range sensor can be extracted from its range image. The challenge here is to extract these features reliably and quickly from range data that may be incomplete and noisy. Previous range image processing (31,41,42) has extracted the following features:

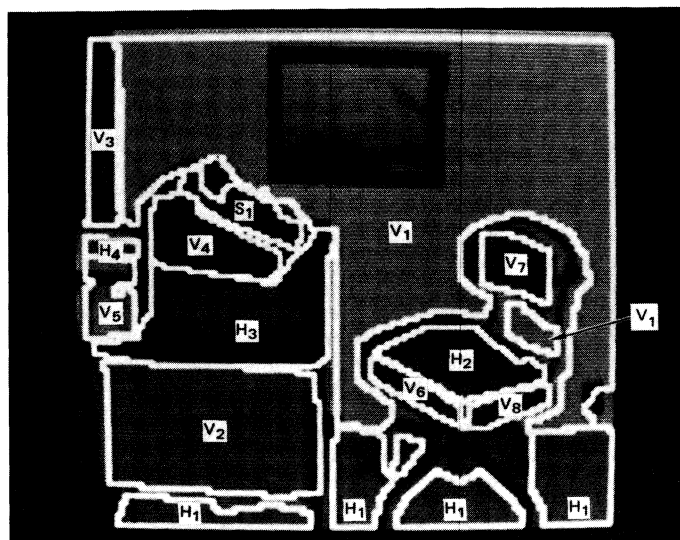
Jump edges, the boundaries between occluding and occluded surfaces (as seen by the range sensor), which are characterized by range discontinuities. Jump edges constitute the portion of the occluding object outline where no contact is made with any other object.

Convex or concave edges, which are characterized by discontinuity in the range gradient.

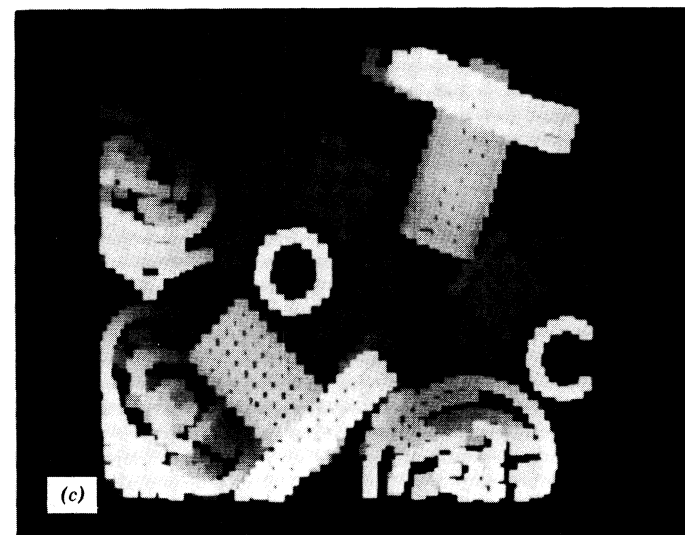
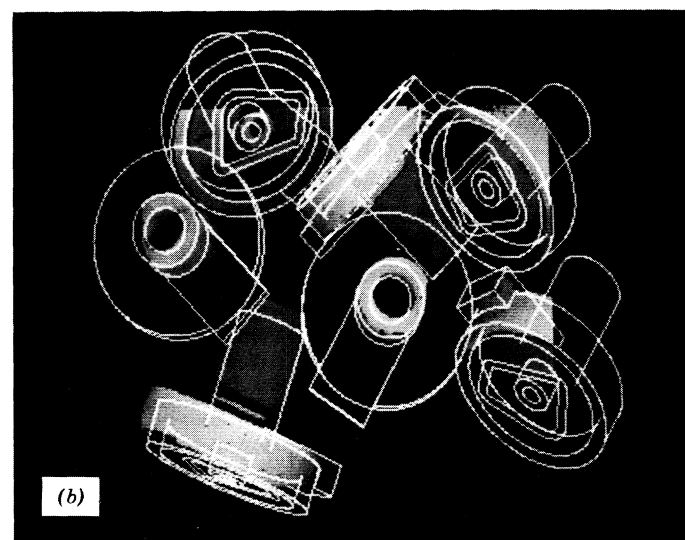
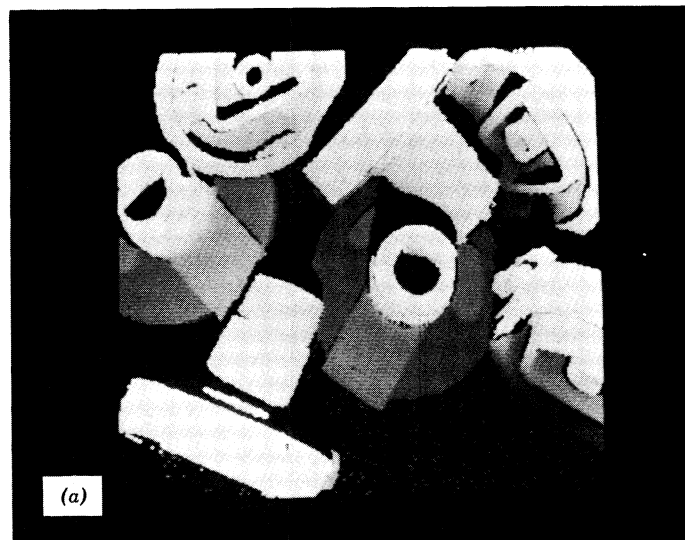
Planar surfaces, classified into horizontal, vertical, and slanted ones (an example is shown in Fig. 10).

More recently, Bolles and Horaud (43) developed a system for determining the location (position and orientation) of each visible part in a bin of jumbled parts of the same type. The system generates a hypothesis about the location of each part by extracting 3-D edge features of three types (straight dihedral, circular dihedral, and straight tangential) from range data and matching a sufficient number of distinctive features to those in the part's model. For example, Figure 11a depicts the measured height of seven castings; higher points are brighter. The system analyzes these range data, extracts 3-D edge features, and hypothesizes the casting locations, as shown in Figure 11b. Next, the resulting hypothesis of patches is verified by predicting the range values on the object's surface and comparing them to the measured data; see Figure 11c. Furthermore, the system locates the parts on top of the pile so that a robot hand can pick them up.

**Generalized Cylinder Surfaces.** A generalized cylinder (qv) is the volume swept by a 2-D region moving along and normal to an axis (straight or curved) in a 3-D space.



**Figure 10.** Office scene partitioned into planar surfaces: H, Horizontal; V, Vertical; S, Slanted. (Source: Ref. 80.)



**Figure 11.** Locating castings in a bin. (a) Measured height data (higher points are brighter). (b) Hypothesized castings. (c) Hypothesis verification. (Source: Courtesy of R. C. Bolles and P. Horaud, SRI International.)

Surfaces of generalized cylinders were used to represent a manufactured doll (44) and other complex objects (45). Generalized cylinder surfaces were extracted from range data (obtained by using planar light slices) and matched with their models.

Contour lines are the intersections of a set of known planes with a 3-D object of an arbitrary shape. The planes may be parallel and equidistant (as in topographical maps), radial with equal angular increments, and the like. Contour lines may be viewed as an extended representation of a generalized cylinder surface formed by sweeping a region of arbitrary shape along an axis that may branch into separate axes. The intersections of a set of light planes with an arbitrary object, used to measure its range image, constitute the visible portions of its contour lines. As such, contour-line representation is compatible with range measurement based on structured light and can be used to inspect the shape and dimensions of 3-D objects (46).

**Volumetric Representations.** Volumetric representations can be used to compute the mass properties of objects as well as their surfaces. Three volumetric representations are distinguished: spatial occupancy, cell decomposition, and constructive solid geometry (38).

A spatial occupancy representation ("voxels") of a solid object is a 3-D array of cells (e.g., cubes); the higher the resolution the larger the required memory space.

A cell decomposition representation is obtained by subdividing a solid object into solid cells (no holes). One such representation is the "Oct-tree," which is obtained by recursive subdivision of the solid until all the cells have elementary shapes.

A solid object may be represented by combining primitive solids, e.g., construction of animals using cylinders.

**Research Issues.** Although impressive progress has been made in edge detection (qv) and relational optical flow (qv) (47), and 3-D vision, these areas require further research. Other research issues in noncontact sensing include parallel computation, object representation, extraction of intrinsic and relational features from surface images, and strategies for matching these features to object models. Different applications have required different sensor types and different object representations. The question is how much of these differences is economical and how much is inherent.

## Contact Sensing

As the name implies, contact sensing requires physical contact with an object whose properties are measured. Contact-sensor signals include tactile, force/torque, temperature, and position; of these, tactile and force/torque sensors are more general.

### Tactile Sensors

**Classification.** Tactile sensors may be classified according to the following criteria:

- resolution—a single element (e.g., a microswitch) vs. an array of elements;
- dynamic range—binary (contact or no contact) vs. "gray-level" (continuous) contact force; and
- directionality—normal vs. tangential force measurement.

**Applications.** A binary microswitch may be used in various "move-till-touch" applications (25), such as reaching a target, preventing collision damage, robot training, object grasping, and dimensional measurement.

Arrays of tactile transducers (binary or gray-scale) mounted on compliant fingers of a robot hand are applicable to object recognition (48), grasping, location, inspection, and slip detection. Rather than being detected, a slip may be prevented by increasing the normal force,  $N$ , just before the sensed tangential force reaches  $\mu_0 N$ , where  $\mu_0$  is the coefficient of static friction between the object and the tactile transducers. Some tactile sensors may also be used to measure the hardness and thermal properties of materials (49); these properties may supplement 3-D features in object recognition.

In a survey conducted by Harmon (50) the following applications for tactile sensing were identified: bin picking, adaptive grasping, assembly, dimensional inspection, shape detection, temperature measurement, tight-part mating, electronic-component insertion, wire-harness construction, limp-material handling, fruit picking, and cow milking.

**Transducers.** Tactile transducers may be based on the following technologies (49,50):

- pressure-sensitive resistivity—simple, inexpensive, and heat resistant but lacking sensitivity;
- semiconductors—small and sensitive but fragile and sensitive to the environment;
- piezoelectric transduction—potentially useful but lacking dc response;
- capacitive transduction—potentially useful but too sensitive to external fields;
- optoelectronic transduction—highly sensitive detection of light whose intensity varies by force-sensitive mechanical means, which are bulky; and
- piezoelectric and pyroelectric transduction—potentially useful for sensing pressure and thermal properties, but sensing them separately is electronically difficult.

A few examples follow.

Hillis (48) developed a  $16 \times 16$ -element tactile sensor consisting of two 16-conductor sheets aligned perpendicularly to each other and separated by a thin elastic medium. The local resistance between each pair of crossing conductors decreases (nonlinearly) as the applied pressure increases. A tactile-pressure image is obtained by applying a voltage to one column conductor at a time and measuring the current flowing in each of the row conductors as they are grounded one at a time. Hillis realized one conductor sheet by an etched flexible printed-circuit board, the other by an anisotropically conductive silicon rubber, and the elastic medium by a nylon woven mesh. Further development of this sensor was terminated due to problems with material manufacturing and robustness (51).

Boie (52) developed a  $6 \times 6$ -element tactile sensor consisting of six conductors etched on an elastic/dielectric layer and six conductors etched normally to the ones above on a flexible printed-circuit board (see Fig. 12). The local capacitance between each pair of crossing conductors increases as the applied pressure increases because the distance between them decreases. This sensor scheme appears encouraging and is being pursued further at MIT (51).

A  $3 \times 6$ -element tactile sensor combining a transducer (pressure-sensitive conductive elastomer) and a VLSI signal

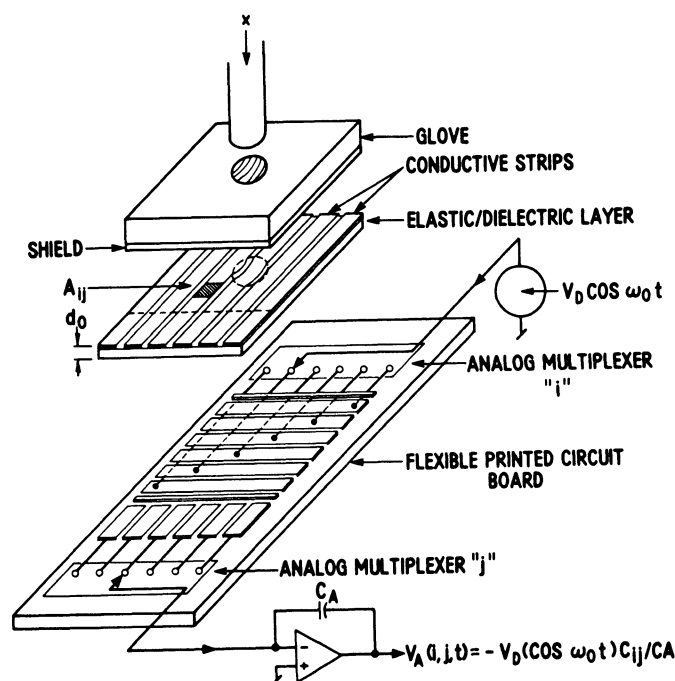


Figure 12. Exploded view of capacitive tactile sensor. (Source: Ref. 52.)

processor has been designed and built by Raibert and Tanner (53) (see Fig. 13). The quality of the transduced data was not good because of the hysteresis and poor mechanical ruggedness of available elastomers and because of the difficulties in placing analog electronics on a chip (54). Raibert (54) has attempted to overcome these problems by developing a scheme for measuring local pressure digitally.

A survey of tactile transducers by Dario and De Rossi (49) includes the following ones: optoelectronic transducers using optical fibers (e.g., at the Jet Propulsion Laboratory in Pasadena, California, and MIT in Cambridge, Massachusetts) and LED/photo detectors (developed at SRI International and manufactured by Lord Corporation); piezoresistive transducers (e.g., at Carnegie-Mellon University in Pittsburgh, Pennsylvania, and LAAS in Toulouse, France); piezoelectric/pyroelectric transducers using polyvinylidene fluoride polymers (PVF2) (e.g., at the University of Pisa, Italy).

Desired performance specifications for tactile sensing depend on the application but, on the average, are as follows (50):

- resolution— $5 \times 10$  to  $10 \times 20$  elements with 1–2 mm spacing (matching the human fingertip);
- sensitivity—1–10 g;
- dynamic range—1000:1;
- response time—1–10 ms;
- nonlinearity is acceptable but hysteresis is not;
- robust sensing “skin” on a compliant material (“flesh”) that covers each finger of a robot hand; and
- local data processing (resulting in a “smart hand”).

Tactile sensors that meet all of these specifications are yet to be developed. Such development could perhaps be helped by studying the mechanisms entailed in human sensing. Most attempts to develop humanlike tactile sensors have failed so far, primarily due to material problems.

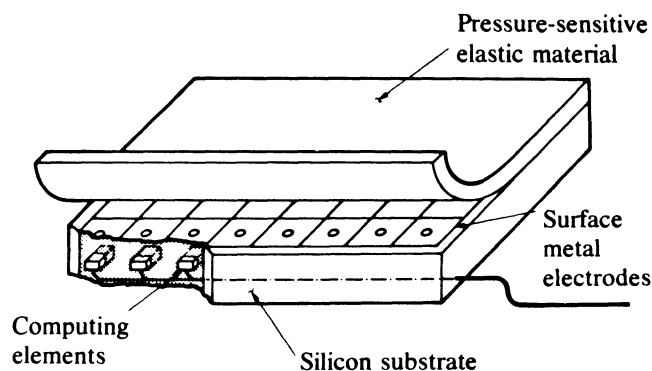


Figure 13. VLSI tactile array sensor. (Source: Ref. 53.)

**Tactile Image Processing.** A tactile image generated by a set of transducer elements that have been activated by contacting an object may be binary or gray-level. Hence, binary or gray-level vision-processing techniques are applicable directly to tactile image processing (e.g., see Ref. 48).

Grimson and Lozano-Perez (55) developed a method for recognition and location of an isolated polyhedral 3-D object with three degrees of freedom based on a set of surface-point positions and normals measured by tactile sensors. The large number of possible interpretations in matching the measured points with modeled object surfaces is reduced considerably by using constraints on the distances between faces, the angles between face normals, and the angles between sensed-point vectors and face normals. This object-matching method was later extended (56) to cases in which the position and orientation of planar-surface patches (or linear segments) are measured by a sensor (e.g., gray-level visual sensor).

A research issue is how to perform image interpretation that is unique to contact sensing. For example, interpreting tactile data associated with object grasping entails friction (which is hard to model) and grasp location. Another issue is dynamic vs. static tactile sensing—how to interpret dynamic tactile data, how the information it provides differs from that of static tactile data, and what the analogy with dynamic and static vision is.

**Applicability of Tactile versus Noncontact Sensing.** Tactile sensing is applicable to object grasping, but its applicability to object recognition, location, and inspection is questionable because these tasks can be done today much more effectively by visual sensing.

Compared with noncontact (vision and range) sensing of a 3-D object, today's tactile sensing has inferior performance specifications, has insufficient techniques for image interpretation, must entail manipulation (e.g., by a robot arm and hand), and may cause some object displacement. On the other hand, tactile sensing is direct, requires no illumination, and can provide information about the object grasping, including slip detection or prevention. In addition, noncontact sensing is inferior if the surface of the object is occluded (e.g., during object grasping), is either too dark (e.g., in deep sea) or too specular (causing transducer blooming or misleading multiple reflections), or is characterized by uniform reflectance despite range variations.

In conclusion, tactile sensing should complement noncontact sensing, not compete with it. Before such sensor integration can be achieved, however, tactile sensing needs to be advanced to a level comparable to that of noncontact sensing.

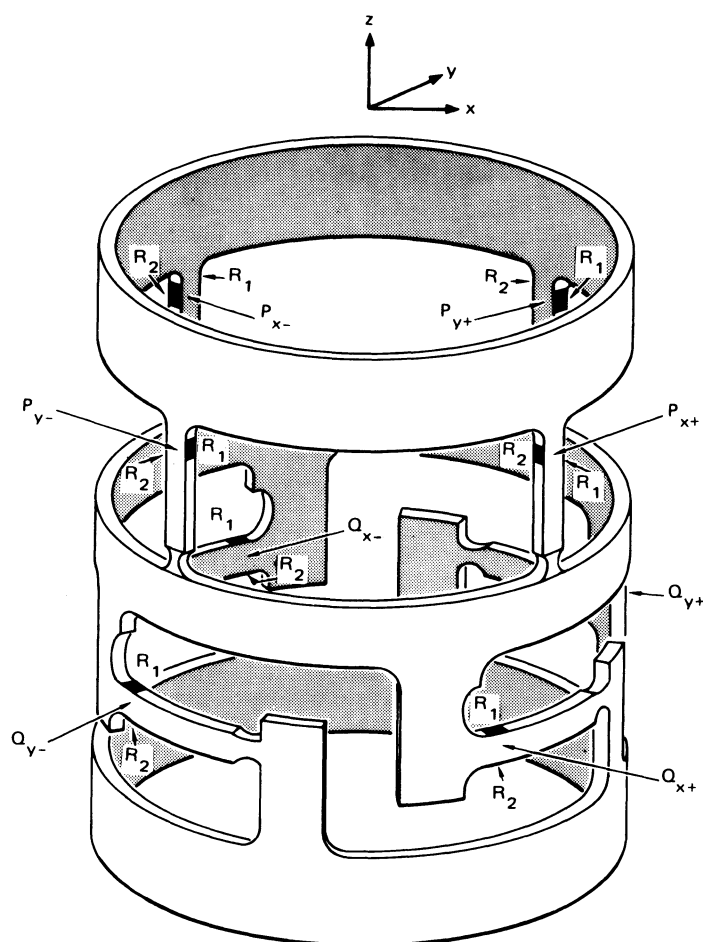


Figure 14. Strain-gauge wrist force sensor.

**Force Sensors.** A six-axis wrist force sensor measures the three components of force and three components of torque acting between the wrist of a robot arm and its end effector which, in turn, exerts the measured force or torque on an object.

Force transduction is achieved by measuring the deflection of compliant sections as a result of the applied force and torque. Force transducers include piezoelectric material and semiconductor strain gauges. The accuracy of force transducers may be increased by mounting them directly on the end-effector fingers rather than on the wrist. Figure 14 shows a strain-gauge wrist sensor developed at SRI (8).

**Research Issues.** Research issues include improving the performance of tactile transducers (see above specifications), analysis of tactile gray-level images (for object handling, recognition, location, and inspection), friction modeling (for slip detection or prevention), and the applicability of tactile sensing with and without noncontact sensing.

Other research issues are effective discrete and continuous sensing, contact location and friction, applicability of contact sensing, combined force/position control, force control of cooperating arms or fingers, performance prediction, and pattern recognition.

### Adaptive Control

**Overview.** The dynamic behavior and positioning accuracy of a robot arm under fixed control vary with the arm configura-

tion and load. An arm with unknown or time-varying parameters could, in principle, be controlled dynamically using adaptive control (see Robot-control systems) by adjusting the parameters according to the position, velocity, and acceleration servo errors. The problem is that the theory for adaptive control of nonlinear, stochastic systems is inadequate (57). Craig (58) divided the dynamic model into two parts—one with known parameters (e.g., link inertia) and the other with a known structure but unknown parameters (e.g., friction and load inertia) that may vary in time. He proposed a scheme using an adaptation law that adjusts the estimated values of the unknown parameters in a closed loop until they converge to values resulting in zero servo errors. The scheme is based on nonlinear equations of motion and uses Liapunov (59) function to guarantee stability but is not fast. Controlling a robot arm dynamically by evaluating its unknown and time-varying parameters in order to minimize the servo errors is a research issue.

Adaptive control of a robot arm may be implemented by using sensors, in particular visual and range sensors, to measure the location (position and orientation) of its end effector relative to a target object, despite measurement delay and noise. Sanderson and Weiss (60) distinguished between two visual feedback representations: a position-based feedback, whose parameters are relative locations, and an image-based feedback, whose parameters are image features. The latter is inherently faster (because it entails no feature-to-space computation delay) but its feedback is nonlinear. They also distinguished between two types of joint-control structures: "look-and-move" and visual tracking.

**Examples.** Adaptive control demonstrations at SRI included the following.

**Compressor-Cover Assembly.** Using global and local features, a compressor housing and its cover were located by a vision subsystem, picked up by a Unimate arm, and placed on an xy table (see Fig. 15). Guided by the vision subsystem, the xy table moved to each of eight positions where an Auto-Place limited-sequence manipulator bolted the cover while the table was free to move slightly to accommodate locational errors. Visual and positional sensing were used to verify this operation (see Ref. 61).

**Tracking.** Visual/range servoing techniques were developed using a projector of a light plane and a TV camera mounted on the end effector of a Unimate arm and applied to

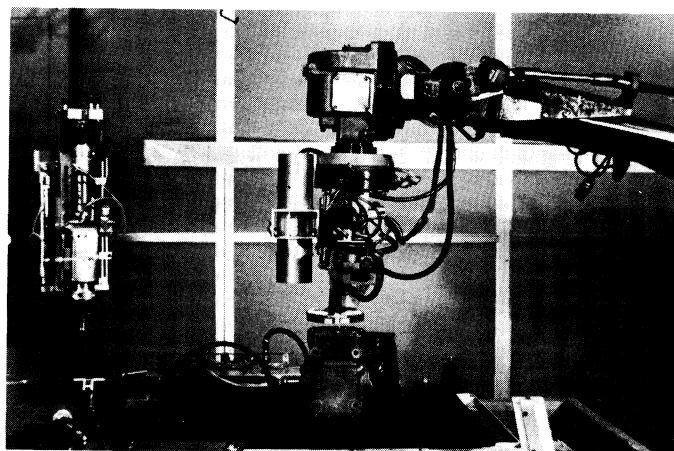
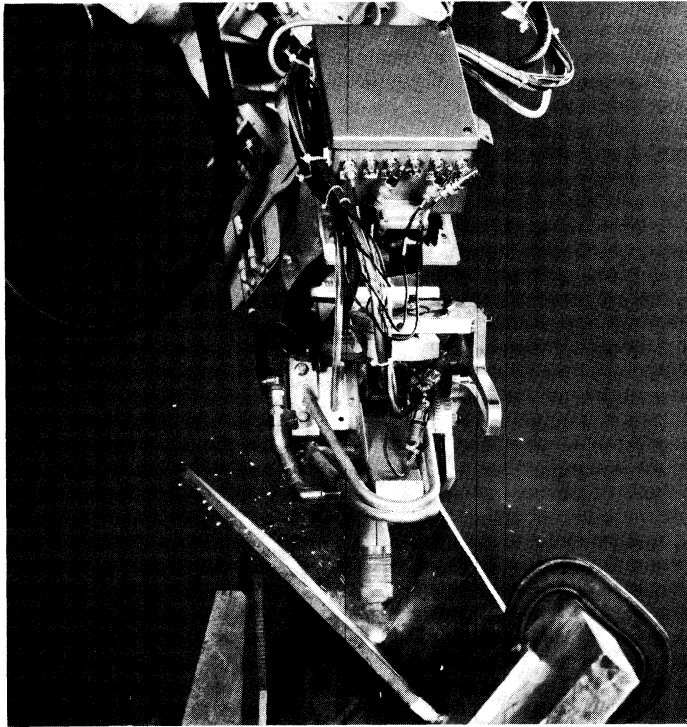


Figure 15. Compressor-cover assembly.





**Figure 16.** Visually guided arc-welding robot. (Source: Courtesy of J. Kremers et al., SRI International.)

simulated spot welding on a moving line and tracking a cornered path in three dimensions (see Ref. 62).

**Arc Welding.** Using a laser scanner range sensor mounted on its wrist, a Cincinnati-Milacron T3 arm was guided by the vision subsystem in one-pass arc welding of different workpiece joints (see Fig. 16) (see Ref. 63).

**Modular Printer-Carriage Assembly.** A nine-part printer carriage was assembled by a modular-assembly station consisting of two PUMA arms with visual and force sensors, respectively, attached to their wrists, a binary vision module, and general-purpose part feeder and assembly fixture on two tables, respectively (see Fig. 17). Binary visual sensing was used to locate plastic rockers and, subsequently, force and "click" sensing were used to verify that they have been snapped properly into a shaft (see Ref. 64).

**Issues.** Some issues entailed in adaptive robot control are as follows.

**Basic Theory.** Developing a general analysis that guarantees stability and high speed in controlling nonlinear, stochastic systems with unknown and varying parameters.

**Sensor Integration.** Selecting the best sensors for a given task and integrating their outputs into the robot control system.

**Branching.** Planning cost-effective conditional branches for robot actions, depending on the sensor signals.

**Execution Speed.** Obtaining and matching high-speed sensory data processing, arm motion, and arm control.

## Robot Programming

**Introduction.** Training today's industrial robot arm is usually done by a human operator who, using a teach box with

push buttons, leads that arm through its task steps and records the location or action of each step. Although popular, this "teaching-by-doing" method may be wasteful (e.g., no useful work during training for batch production), tedious (e.g., training for many computable locations), or inadequate (e.g., training for sensor-guided task steps). These limitations can be partially overcome by using a computer-based textual robot-programming language. Furthermore, a robot-programming language (or an equivalent computer program) is essential for programming a robot system, working alone or with other machines, that consists of manipulators, sensors, and auxiliary devices and is controlled by a hierarchy of distributed computers.

A robot programming-language (qv) is applicable in two programming modes: on-line programming and off-line programming. On-line programming is performed by a programmer who, although sensing and handling real equipment and workpieces, generates a program text that may include manipulator locations (either taught "by doing" or typed in), robot sensing (to overcome uncertainties), and logic or control statements (which usually constitute the bulk of the program). Off-line programming is similar to on-line programming except that the programmer deals with simulated objects (workpieces, sensors, manipulators, and other equipment) rather than real objects.

**Robot-Programming Languages.** The main goal of using a robot-programming language is to facilitate the programming of a robot system for a new task or modification of an old one. To achieve this goal, a robot-programming language provides the user with high-level programming capabilities. These capabilities are implemented by means of a language processor and a robot controller—the processor accepts and checks the user statements and translates them into commands for the controller; the controller then generates lower level commands for the corresponding device (e.g., the trajectories, joint values, and servo commands for the arm joints).

**Current Robot-Programming-Language Capabilities.** Eight commercially available U.S. robot-programming languages have been compared by Gruver et al. (65):

- AL—Stanford University (66),
- AML—IBM Corporation (67),
- HELP—General Electric Company (68),
- JARS—Jet Propulsion Laboratory (69),
- MCL—McDonnell Douglas Corporation (70),
- RAIL—Automatix, Inc. (71),
- RPS—SRI International (72), and
- VAL—Unimation, Inc. (73).

The combined features of these robot languages are classified below into general and robotic programming capabilities (see the above references for individual robot-language capabilities).

**General Programming Capabilities.** General programming capabilities include the following high-level language features:

- Data types**—integer, real, character string, label, and aggregate (an ordered set of data types (67);
- Operations**—Arithmetic, relational, logical, assignment, etc.;



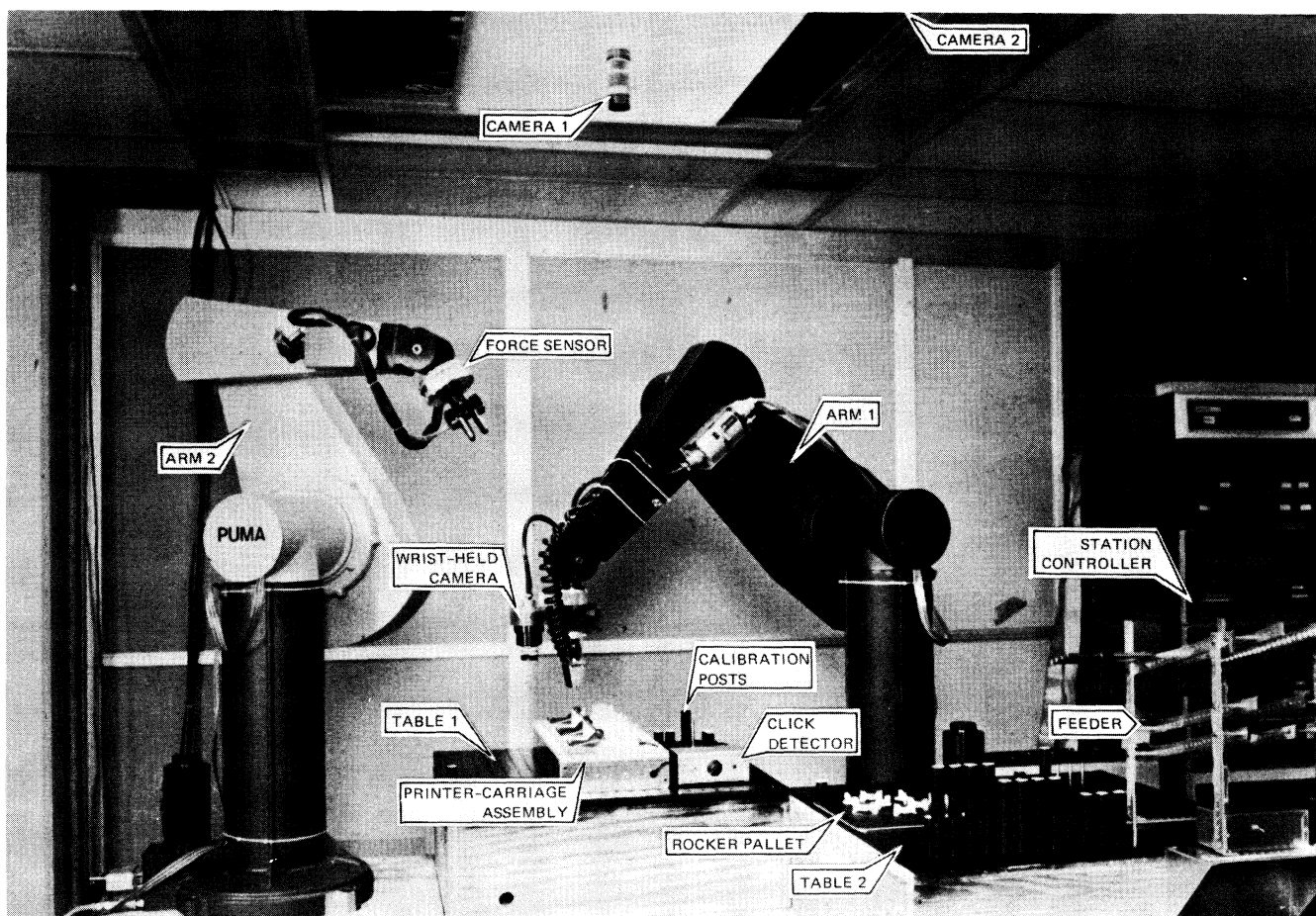


Figure 17. Modular assembly station.

*Control expressions*—block structure (BEGIN-END), branching (GOTO), conditional branching (IF-THEN; IF-THEN-ELSE-), continuing (WHILE-DO-), and looping (DO-UNTIL-);

*Subroutines and functions*—library (compiled) and user-generated (interpreted);

*Interactive support modules*—text editing, hot editing (run, stop, edit, and continue to run a program), compiling, interpreting, graphic simulation, etc.; and

*Debugging features*—break points, tracing, and single-stepping.

*Robotic Programming Capabilities.* Robotic programming capabilities, invoked by declarations or commands, are usually built into the robot language; in a few cases, however, they are incorporated into the language by interfacing with external modules (e.g., RPS interfaces with the PUMA VAL (Unimation, Inc.) controller and the SRI and MIC vision modules). These capabilities are classified below according to the nature of the robot functions:

*Geometric data types*—vector, displacement, rotation, frame, transform, and path of points;

*Motion of arm end effector*—specified joint(s), interpolated joints (between two points), straight line (to a given destination), straight line via a given point, continuous path (through given points), sawtooth weaving superimposed on

a continuous path, specification of speed and acceleration or deceleration, departure and approach, etc.;

*Vision*—picture taking, binary feature extraction, silhouette-based object recognition and location, adjustment of thresholds and windows, and gray-level feature extraction (e.g., histograms);

*Servoing with sensory feedback*—visual sensing, limit switch, and force/torque sensing; and

*Multiprocessing*—simultaneous control of multiple arms, sensors, machines, and other devices in a manufacturing cell.

*Limitations and Issues.* Current robot-programming languages are handicapped by the following limitations and issues (74,75):

*System Integration.* Robot-programming languages, currently capable of controlling a single arm, should be able to concurrently control a flexible workcell comprised of two or more arms, machines, vision modules, other sensors, and auxiliary equipment, and be integrable into the factory CAD/CAM system.

*Task-Level Commands.* Robot-programming languages, currently including only manipulator-level commands, should also include higher, task-level commands to simplify workcell programming for more complex tasks.

*Flexibility and Intelligence.* Robot-programming languages should be flexible and intelligent enough to handle unpredict-

able situations, e.g., be able to verify that specified trajectories are collision-free.

**Hardware Dependency.** Robot-programming-language software is closely tied to specific computer hardware; although achieving higher speed, this dependency hinders computer portability.

**Language standardization.** A commercial robot-programming language is proprietary with its vendor and depends heavily on the kinematics of the arm it controls; hence, users of different arms must program them in several languages. This problem could be solved by having a standard, arm-independent language processor that interfaces with each arm-specific controller that serves the arm joints.

**Marketing Considerations.** Robot-arm marketing appears to be the major incentive for developing a robot programming language. This factor may oppose efforts to standardize robot-programming languages unless one robot manufacturer dominates the market.

**User/Language Compatibility.** The spectrum of users who match the complexity of a given robot-programming language is narrow. Developing robot-programming languages that match a wide spectrum of users, from a factory worker to a computer scientist, is an issue. Along this line, one may question the utility of a robot-programming language that is restrictive when used by a sophisticated programmer who requires programming flexibility. An alternative approach is to utilize the general programming capabilities of a high-level language (such as C, Pascal, or Ada) and, with an appropriate computer operating system, develop hierarchies of robot-function subroutines that constitute robot-programming capabilities. This approach could reduce the cost and enhance the portability of the robot software as well as improve programming flexibility.

**Off-Line Programming.** Off-line programming is performed by a programmer who generates a program text that may include simulated manipulator locations, simulated robot sensing, and logic or control statements.

**Advantages and Disadvantages.** Compared with on-line programming, off-line programming has the following advantages.

**No production stoppage.** Production need not be stopped to make real robotic equipment available for programming. This is a major cost-reduction factor.

**Safety.** The danger of harm to the programmer, the equipment, and the workpieces during programming is eliminated.

**Early Programming.** Programming can begin before workpieces, robots, machines, fixtures, and other equipment arrive and, hence, is not susceptible to delays caused by late deliveries.

**Product Redesign.** Any mistake in the design of a workpiece or a product can be detected and corrected before it is produced.

**Sensitivity Verification.** The generated program can be verified under exhaustive conditions that can be simulated but not realized experimentally. For the same reason, determining the sensitivity of operation to variation of parameters must be based on simulation.

On the other hand, the disadvantages of off-line programming are described below.

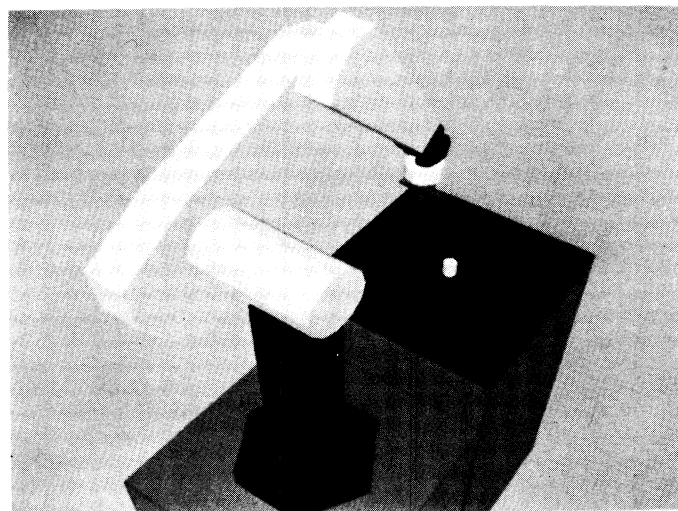
**Model Limitations.** The accuracy of simulating a robot system depends on the authenticity of its model. Complex phenomena that are hard to model precisely will result in simulation errors.

**Modeling Cost.** Unlike on-line programming, simulation in off-line programming entails modeling cost.

**3-D Geometric Models.** Three types of geometric models of 3-D objects are distinguished:

1. *wire-frame models*—representing objects by their vertices and edges; These models are simple and fast but are hard to visualize (especially if hidden lines are not eliminated), are useless for collision detection, and may be ambiguous (see Representation, wire-frame);
2. *surface models*—representing objects by their surfaces; the above wire-frame model drawbacks are eliminated by using surface models, but they are slow and provide no information about the mass, inertia, and stability of objects; and
3. *solid models*—representing objects by their volumes (including surfaces); most of the above drawbacks are eliminated by using solid models, but they are slow.

**Simulation.** A graphic simulation system uses a 3-D model to generate static or dynamic displays of objects and workcell layout. Such a system enables a programmer to view workcell objects from different directions (using perspective transformations) with variable magnification (e.g., zooming), as if he or she were a "flying eye" observing a real workcell. Using this facility, an off-line programmer interactively writes and debugs the program steps for a given task, observes the animation of the operation, and when satisfied with the results, downloads the program onto the (real) workcell controller and runs it. For example, Figure 18 shows a display (generated on a Silicon Graphics IRIS 2400 system) of a simulated PUMA arm with a two-finger hand mounted on a wrist force sensor, a table, and a cylindrical part on it. A simple task was programmed off-line and executed by a real, calibrated workcell: a vision module (simulated but not shown in Fig. 18) located the part and the arm, guided by this information, picked up the part and moved it to its destination.



**Figure 18.** Graphic simulation in off-line programming of a robot system. (Source: Courtesy of R. C. Smith, SRI International.)

**CAD/CAM Database.** Ultimately, the geometric and non-geometric representations of all the equipment and workpieces in a factory will be stored in a computer, and these databases will be applied to CAD (qv) and computer-aided manufacturing (CAM) (see Computer-integrated manufacturing). Currently, however, a limited amount of such a database exists in certain industries (e.g., some aerospace and automotive industries), and its use is limited to static and dynamic graphic display in the design stage and to programming numerically controlled (NC) part machining in the manufacturing stage.

**Research Issues.** Among the research issues related to off-line programming are the following ones:

*3-D object modeling*—increasing the speed of processing surface and solid models both algorithmically and by hardware; modeling flexible objects; locational uncertainty;

*CAD/CAM*—tolerances; path planning and collision avoidance; task-level robot functions acting on workpieces under a range of constraints; near optimal robot selection and workcell-layout design; nongeometric workpiece and equipment representations; standardizing robot tools and types; rules for manufacturing processes (fabrication, handling, inspection, and assembly); integration of different CAD/CAM systems; and

*simulation*—efficient user interface; robot sensing simulation; robot reach; robot dynamics; static and dynamic process simulation.

## Manufacturing Process Planning

Manual robot-system programming entails a major effort even if facilitated by a robot-programming language or done off-line. Graphic simulation may help specify robot locations, but not logic or control statements, which usually constitute most of the program. Developing a task-oriented programming language will mitigate this problem, but a long-term solution is automatic robot-system planning and programming.

Consider a programmable assembly workcell that is controlled by a hierarchy of distributed computers and includes manipulators, sensors, part feeders, and auxiliary devices. If instructed to assemble a particular product, the workcell must be provided with a workable plan for the assembly process. Consider how such a plan can be generated cost-effectively in a future factory equipped with robotic CAD/CAM systems.

The output of the CAD system for a given product will include two major components: the design of the product parts and their attachment sequence. This information is used to plan the robotic assembly operations, each of which includes actions and verifications of equipment, parts, and the actions themselves. The CAD output can be generated interactively by a human designer without much difficulty. In contrast, planning the details of the actions and, especially, the verification of each assembly operation is very tedious and prone to human errors. Hence, there is a need to automate planning of robotic assembly processes (and, for similar reasons, other manufacturing processes).

Process planning (qv) depends on the "state of the process world," which describes the equipment, workpieces, and actions vs. time, and is part of the process "knowledge base." Few automatic planners using AI techniques have been developed to date, and these planners [e.g., DEVISER (76)] are

based on the assumption that the state of the process world is known exactly at the planning time. In an unstructured world, where this assumption is invalid, the state of the process world should be determined by robot sensors.

Development of a sensor-based planner and execution monitor using AI techniques was begun by Cheeseman at SRI (2). The planner will include a knowledge base ("expert system"), which consists of a world model (effectors, sensors, and other devices), process rules, and sensor-selection rules, as well as a plan generator which plans the action and checking steps for a given robot process. The generated plan will be sent to the execution controller, which will command and coordinate the process execution. As the process world changes, its state will be updated by the robot sensors; this information is required to execute and monitor the plan actions. The highlights of the sensor-based planner are described below.

The plan will call for the use of sensors at run time to fill in unknown information of two types:

enabling execution of actions in the plan, e.g., visually servoing an arm, and

monitoring the plan execution, i.e., verifying its actions.

The planner will select the best sensor(s) for a given task on the basis of rules and estimated locational errors of objects, effectors, and sensors.

Conditional planning will provide alternative branches for all possible sensor outputs, followed by immediate rejoining of the branches to simplify the plan (i.e., prevent branch "bushiness").

Multiple sensors will be used to combine sensory information if a single sensor is not sufficient. Fixed sensors (e.g., mounted on the ceiling) will be distinguished from manipulated ones (e.g., mounted on robot end effectors) because the latter entail actions, in addition to information gathering, which must be planned.

Error detection and correction will be implemented. If a failure is detected, the process will be halted and, if possible, replanned locally to minimize production down time.

Research issues in planning include the following: a formal theory and representation of knowledge (e.g., geometry, uncertainty, temporal relationships, and condition monitoring); planning collision-free paths, fine-motion sensing, part mating, error detection and correction, and multiple robot cooperation; real-time planning and replanning; planning for multiple agents (distributed AI); and planner learning.

These research issues are extremely difficult to solve. Despite the importance of automatic planning, its implementation will take many years. Eventually, however, automatic robot-system planning will be an essential component of the factory.

## BIBLIOGRAPHY

1. J. Jablonowski and J. W. Posey, Robotics Terminology, in S. Y. Nof (ed.), *Handbook of Industrial Robotics*, Wiley, New York, pp. 1271–1303, 1985.
2. D. Nitzan, C. Bavrouil, P. C. Cheeseman, and R. C. Smith, Use of Sensors in Robot Systems, *Proceedings of the 1983 IEEE International Conference on Advanced Robotics*, Tokyo, Japan, September 12–13, 1983, pp. 123–132.
3. K. Sadamoto, *Robots in the Japanese Economy*, Survey Japan, Tokyo, 1981.

4. Y. Umetani and K. Yonemoto, Japan Robotics Research for the Next Generation, *Proceedings of the 1983 IEEE International Conference on Advanced Robotics*, Tokyo, Japan, September 12–13, 1983, pp. 3–20.
5. J. F. Engelberger, *Robotics in Practice*, Kogan Page, London, in association with Avebury, Amersham, 1980.
6. D. Nitzan and C. A. Rosen, "Programmable industrial automation," *IEEE Trans. Comput.* C-25(10), 1259–1270 (December 1976).
7. J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *J. Appl. Mechan.* (ASME) 215–221 (June 1955).
8. C. A. Rosen and D. Nitzan et al., Exploratory Research in Advanced Automation, Second Report, NSF Grant GI-38100X1, Stanford Research Institute, Menlo Park, CA, August 1974.
9. R. P. Paul, *Robot Manipulation: Mathematics, Programming, and Control*, MIT Press, Cambridge, MA, 1981.
10. R. P. Paul, Manipulator Path Control, *Proceedings of the International Conference on Cybernetics and Society*, San Francisco, CA, (September 1975), pp. 147–152.
11. C. A. Rosen and D. Nitzan et al., Machine Intelligence Research Applied to Industrial Automation, Sixth Report, NSF Grant APR75-13074, Stanford Research Institute, Menlo Park, CA, November 1976.
12. R. Featherstone, "The calculation of robot dynamics using articulated-body inertias," *Int. J. Robot. Res.* 2(1), 13–30 (Spring 1983).
13. J. M. Hollerbach, "A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity," *IEEE Trans. Syst. Man Cybernet.* SMC-10(11), 730–736 (1980).
14. J. Y. S. Luh et al., "On-line computational scheme for mechanical manipulators," *Trans. ASME, J. Dyn. Sys. Meas. Contr.* 102(2), 69–76 (1980).
15. W. M. Silver, "On the equivalence of Lagrangian and Newton–Euler dynamics for manipulators," *Int. J. Robot. Res.* 1(2), 60–70 (Summer 1982).
16. T. R. Kane and D. A. Levinson, "The use of Kane's dynamical equations in robotics," *Int. J. Robot. Res.* 2(3), 3–21 (Fall 1983).
17. T. Okada and S. Tsuchiya, On a Versatile Finger System, *Proceedings of the Seventh International Symposium on Industrial Robots*, Tokyo, Japan, October 19–21, 1977, pp. 345–352.
18. J. K. Salisbury and J. J. Craig, "Articulated hands: Force control and kinematic issues," *Int. J. Robot. Res.* 1(1), 4–17 (Spring 1982).
19. S. Drake, Using Compliance in Lieu of Sensory Feedback for Automatic Assembly, Charles Stark Draper Laboratory, Cambridge, MA, Report T-657, September 1977.
20. T. Iwamoto, H. Yamamoto, and K. Honma, Transformable Crawler Mechanism with Adaptability to Terrain Variations, *Proceedings of the 1983 IEEE International Conference on Advanced Robotics*, Tokyo, Japan, September 12–13, 1983, pp. 285–291.
21. R. B. McGhee, Vehicular Legged Locomotion, in G. N. Saridis (ed.), *Advances in Automation and Robotics*, Jai, Greenwich, CT, 1983.
22. M. H. Raibert and I. E. Sutherland, "Machines that walk," *Sci. Am.* 248(2), 44–53 (January 1983).
23. T. G. Bartholet, The First "Functionoid" Developed by Odetics, Inc., *Proceedings of the 1983 IEEE International Conference on Advanced Robotics*, Tokyo, Japan, September 12–13, 1983, pp. 293–298.
24. B. Raphael, *The Thinking Computer: Mind Insider Matter*, W. H. Freeman, San Francisco, CA, 1976.
25. D. Nitzan, Assessment of Robotic Sensors, *Proceedings of the First International Conference on Robot Vision and Sensory Controls*, Stratford-Upon-Avon, U.K., April 1981, pp. 1–11.
26. J. L. Mundy, The Limits of Accuracy for Range Sensing, in D. Nitzan and R. C. Bolles (eds.), *Workshop on Intelligent Robots: Achievements and Issues*, SRI International, Menlo Park, CA, pp. 109–133, 1985; B. K. P. Horn, "Shape from Shading," in P. H. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, NY, 1975.
27. R. A. Jarvis, "A Perspective on Range Finding Techniques for Computer Vision," *IEEE Trans. PAMI* PAMI(2), 122–139 (March 1983).
28. D. Nitzan, R. C. Bolles, J. H. Kremers, and P. G. Mulgaonkar, 3-D Vision for Robot Applications, *Proceedings of the NATO Workshop on Knowledge Engineering for Robotic Applications*, Maratea, Italy, May 12–16, 1986.
29. M. D. Altschuler et al., "The Numerical Stereo Camera," *Proceedings of the Society for Photo-Optical Instrumentation Engineers Conference on 3-D Machine Perception*, Vol. 283, SPIE, Bellingham, WA, pp. 15–24, 1981.
30. A. R. Johnston, Infrared Laser Rangerfinder, Report NPO-13460, Jet Propulsion Laboratory, NASA, New Technology, Pasadena, CA, August 1973.
31. D. Nitzan, A. Brain, and R. Duda, The Measurement and Use of Registered Reflectance and Range Data in Scene Analysis, *Proc. IEEE* 65(2), 206–220 (February 1977).
32. F. E. Goodwin, Coherent Laser Radar 3-D Vision Sensor, SME Technical Paper MS85–1005, Society of Manufacturing Engineers, Dearborn, MI, 1985.
33. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
34. E. C. Hildreth, "Edge detection for computer vision system," *Mechan. Eng.* 48–53 (August 1982).
35. G. J. Gleason and G. J. Agin, A Modular Vision System for Sensor-Controlled Manipulation and Inspection, *Proceedings of the Ninth International Symposium and Exposition on Industrial Robots*, Washington, DC, March 1979, pp. 57–70.
36. R. C. Bolles and R. A. Cain, "Recognizing and locating partially visible objects: The local-feature-focus method," *Int. J. Robot. Res.* 1(3), 57–82 (Fall 1982).
37. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, New York, 1976.
38. D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, 1982.
39. M. Miyagawa et al., Flexible Vision System "Multi-Window" and This Application, *Proceedings of the 1983 International Conference on Advanced Robotics*, Tokyo, Japan, September 12–13, 1983, pp. 171–178.
40. S. Barnard, Automated Inspection Using Gray-Scale Statistics, *Proceedings of the First AAAI Conference*, Stanford University, Stanford, CA, August 19–21, 1980, pp. 49–52.
41. D. Nitzan, Scene Analysis Using Range Data, AI Center Technical Note 69, Stanford Research Institute, Menlo Park, CA, August 1972.
42. R. O. Duda, D. Nitzan, and P. Barret, "Use of range and reflectance data to find planar surface regions," *Proc. IEEE Trans. Patt. Anal. Mach. Intell.* PAMI-1(3), 259–271 (July 1979).
43. R. C. Bolles and P. Horaud, "3DPO: A three-dimensional part orientation system," *Int. J. Robot. Res.* 5(3), (1986).
44. G. J. Agin and T. O. Binford, Computer Description of Curved Objects, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford, CA, August 1973, pp. 629–640.
45. R. Nevatia and T. O. Binford, Structured Descriptions of Complex Objects, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford, CA, August 1973, pp. 641–647.
46. P. G. Mulgaonkar, D. Nitzan, C. K. Cowan, and A. Bavarsky, Inspection of Three-Dimensional Objects, *Proceedings of the Twelfth NSF Conference on Production Research and Technology*, University of Wisconsin, Madison, WI, May 1985, pp. 431–435.

47. E. C. Hildreth, *The Measurement of Visual Motion*, MIT Press, Cambridge, MA, 1983.
48. W. D. Hillis, "A high-resolution image touch sensor," *Int. J. Robot. Res.* 1(2), 33-44 (Summer 1982).
49. P. Dario and D. De Rossi, Tactile Sensors and the Gripping Challenge, *Proc. IEEE Spectr.* 22(8), 46-52 (August 1985).
50. L. D. Harmon, "Automated tactile sensing," *Int. J. Robot. Res.* 1(2), 3-32 (Summer 1982).
51. J. M. Hollerbach, Tactile Sensors and Interpretation of Contact Features, in D. Nitzan and R. C. Bolles (eds.), *Workshop on Intelligent Robots: Achievements and Issues*, SRI International, Menlo Park, CA, 1985, pp. 143-152.
52. R. A. Boie, Capacitive Impedance Readout Tactile Image Sensor, *Proceedings of the 1984 IEEE International Conference on Robotics*, Atlanta, GA, March 13-15, 1984, pp. 370-378.
53. M. H. Raibert and J. E. Tanner, "Design and implementation of a VLSI tactile sensing computer," *Int. J. Robot. Res.* 1(3), 3-18 (Fall 1982).
54. M. H. Raibert, An All Digital VLSI Tactile Array Sensor, *Proceedings of the 1984 IEEE International Conference on Robotics*, Atlanta, GA, March 13-15, 1984, pp. 314-319.
55. W. E. L. Grimson and T. Lozano-Perez, Model-Based Recognition and Location from Tactile Data, *Proceedings of the 1984 IEEE International Conference on Robotics*, Atlanta, GA, March 13-15, 1984, pp. 248-255.
56. W. E. L. Grimson and T. Lozano-Perez, Recognition and Location of Overlapping Parts from Sparse Data in Two and Three Dimensions, *Proceedings of the IEEE International Conference on Robotics and Automation*, St. Louis, MO, March 25-28, 1985, pp. 61-66.
57. A. C. Sanderson and L. E. Weiss, Adaptive Control of Sensor-Based Robotic Systems, in D. Nitzan and R. C. Bolles (eds.), *Workshop on Intelligent Robots: Achievements and Issues*, SRI International, Menlo Park, CA, 1985, pp. 185-205.
58. J. J. Craig, An Adaptive Algorithm for the Computed Torque Method of Manipulator Control, in D. Nitzan and R. C. Bolles (eds.), *Workshop on Intelligent Robots: Achievements and Issues*, SRI International, Menlo Park, CA, 1985, pp. 171-184.
59. W. Hahn, *Theory and Application of Liapunov's Direct Method*, Prentice-Hall, Englewood Cliffs, NJ, 1963.
60. A. C. Sanderson and L. E. Weiss, Adaptive Visual Servo Control of Robots, in A. Pugh (ed.), *Robot Vision*, IFS Publications, U.K., 1983.
61. D. F. McGhie and J. W. Hill, Vision Controlled MS78-685 Subassembly Station, *SME Robots III Conference*, Technical Paper MS78-685, Chicago, IL, November 7-9, 1978.
62. G. J. Agin, Real Time Control of a Robot with a Mobile Camera, *Proceedings of the Ninth International Symposium and Exposition on Industrial Robots*, Washington, DC, March 1979, pp. 233-246.
63. J. H. Kremers et al., Development of a Machine-Vision Based Robotic Arc-Welding System, *Proceedings of the Thirteenth International Symposium on Industrial Robots and Robots*, Vol. 7, Chicago, IL, April 18-21, 1983, pp. 14.19-14.33.
64. R. C. Smith and D. Nitzan, A Modular Programmable Assembly Station, *Proceedings of the Thirteenth International Symposium on Industrial Robots and Robots*, Vol. 7, Chicago, IL, April 18-21, 1983, pp. 5.53-5.75; reprinted in W. B. Heigenbotham (ed.), *Programmable Assembly*, Springer-Verlag, Berlin, 1984, pp. 197-216.
65. W. A. Gruver, B. I. Soroka, J. J. Craig, and T. L. Turner, Evaluation of Commercially Available Robot Programming Languages, *Proceedings of the Thirteenth International Symposium on Industrial Robots and Robots*, Vol. 7, Chicago, IL, April 18-21, 1983, pp. 12.58-12.68.
66. S. Mujtaba and R. Goldman, *AL User's Manual*, Third Edition, Report No. STAN-CS-81-889, Stanford University, Stanford, CA, December 1981.
67. R. H. Taylor, P. D. Summers, and J. M. Meyer, "AML: A manufacturing language," *Int. J. Robot. Res.* 1(3), 19-41 (Fall 1982).
68. General Electric, *Allegro Operator's Manual*, A12 Assembly Robot, General Electric Company, Bridgeport, CT, 1982.
69. J. J. Craig, JARS: JPL Autonomous Robot System, Unpublished Report, Jet Propulsion Laboratory, Pasadena, CA, 1980.
70. B. O. Wood and M. A. Fugelso, MCL, The Manufacturing Control Language, *Proceedings of the Thirteenth International Symposium on Industrial Robots and Robots*, Vol. 7, Chicago, IL, April 18-21, 1983, pp. 12.84-12.96.
71. Automatix, *RAIL Software Reference Manual (ROBOVISION and CYBERVISION)*, Rev. 3.0, MN-RB-07, Automatix, Burlington, MA, January 1982.
72. W. T. Parks, The SRI Robot Programming system (RPS), *Proceedings of the Thirteenth International Symposium on Industrial Robots and Robots*, Vol. 7, Chicago, IL, April 18-21, 1983, pp. 12.21-12.41.
73. Unimation, User's Guide to VAL, Version 12, No. 398-H2A (June 1980); VAL Univision Supplement, Version 13 (VSN), 2nd ed., Unimation Inc., Danbury, CT, July 1981.
74. B. I. Soroka, "What Can't Robot Languages Do?" *Proceedings of the Thirteenth International Symposium on Industrial Robots and Robots*, Vol. 7, Chicago, IL, April 18-21, 1983, pp. 12.1-12.8.
75. R. Goldman, Programming Languages for Robots, in D. Nitzan and R. C. Bolles (eds.), *Workshop on Intelligent Robots: Achievements and Issues*, SRI International, Menlo Park, CA, 1985, pp. 231-236.
76. S. A. Vere, Planning in Time: Windows and Durations for Activities and Goals, Jet Propulsion Laboratory Report, California Institute of Technology, Pasadena, CA, November 1981.
77. R. U. Ayres and S. M. Miller, *Robotics: Applications and Social Implications*, Ballinger Publishing Company, Cambridge, MA, 1983.
78. R. H. Taylor, "Planning and Execution of Straight Line Manipulator Trajectories," *IBM J. Res. Develop.* 23(4), 253-264 (July 1979).
79. B. K. P. Horn, *Robot Vision*, MIT Press, Cambridge, MA, 1986.
80. R. Duda, D. Nitzan, and P. Barrett, *IEEE Trans. PAMI* PAMI-1(3), 259-271 (July 1979).

## General References

### Books

- J. K. Aggarwal, R. O. Duda, and A. Rosenfeld (eds.), *Computer Methods in Image Analysis*, IEEE Press, New York, 1977.
- I. Aleksander, *Artificial Vision for Robots*, Chapman & Hall, New York, 1984.
- I. Aleksander (ed.), *The World Yearbook of Robotics Research and Development*, Kogan Page, London, 1985.
- B. G. Batchelor et al., (eds.), *Automated Visual Inspection*, IFS Publications, UK and North Holland, 1985.
- G. Beni and S. Hackwood (eds.), *Recent Advances in Robotics*, John Wiley & Sons, New York, 1985.
- J. M. Brady (ed.), *Computer Vision*, North Holland, Amsterdam, 1981.
- J. M. Brady et al. (eds.), *Robot Motion: Planning and Control*, MIT Press, Cambridge, MA, 1982.
- J. M. Brady and R. Paul, *Robotics Research: The First International Symposium*, MIT Press, Cambridge, MA, 1984.
- P. Coiffet, *Robot Technology*, A series of seven volumes, Prentice-Hall, Englewood Cliffs, NJ, 1981.
- J. J. Craig, *Introduction to Robotics: Mechanics and Control*, Addison-Wesley, Reading, MA, 1986.



- A. J. Critchlow, *Introduction to Robotics*, Macmillan, New York, 1985.
- G. G. Dodd and L. Rossol (eds.), *Computer vision and Sensor-Based Robots*, Plenum Press, New York, 1979.
- R. C. Dorf, *Robotics and Automated Manufacturing*, Reston Publishing Co., Reston, VA, 1983.
- W. B. Gevartner, *Intelligent Machines—An Introductory Perspective of Artificial Intelligence and Robotics*, Prentice-Hall, Englewood Cliffs, NJ, 1985.
- M. P. Groover and E. W. Zimmers, Jr., *CAD/CAM—Computer-Aided Design and Manufacturing*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
- A. R. Hansen and E. M. Riseman, *Computer Vision Systems*, Academic Press, New York, 1978.
- W. B. Heginbotham (ed.), *Programmable Assembly*, IFS Publications, UK and Springer-Verlag, Berlin, 1984.
- V. D. Hunt, *Smart Robots*, Chapman & Hall, New York, 1985.
- C. G. S. Lee, R. C. Gonzales, and K. S. Fu, *Tutorial on Robotics*, Second Edition, Computer Society Press, Silver Spring, MD, 1986.
- M. D. Levine, *Vision in Man and Machine*, McGraw-Hill, New York, 1985.
- D. Marr, *Vision*, W. H. Freeman, San Francisco, 1982.
- M. E. Mortenson, *Geometric Modeling*, John Wiley & Sons, New York, 1985.
- R. Nevatia, *Machine Perception*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
- W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill, New York, 1979.
- S. Y. Nof (ed.), *Handbook of Industrial Robots*, John Wiley & Sons, New York, 1985.
- Y. C. Pao, *Elements of Computer-Aided Design and Manufacturing*, John Wiley & Sons, New York, 1984.
- T. Pavlidis, *Structural Pattern Recognition*, Springer-Verlag, Berlin, 1977.
- W. Pratt, *Digital Image Processing*, John Wiley & Sons, New York, 1978.
- A. Pugh, *Robot Vision*, IFS Publications, UK and Springer-Verlag, New York, 1983.
- A. Pugh, *Robot Sensors, Vol. 1-Vision; Vol. 2-Tactile and Non-Vision*, IFS Publications, UK and Springer-Verlag, Berlin, 1986.
- P. Ranky, *The Design and Operation of Flexible Manufacturing Systems*, IFS Publications, UK and North Holland Publishing Company, New York, 1983.
- W. G. Rohm, "A Remote Promise," *Infosystems*, 52–56 (Sept. 1986).
- A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, New York, 1976.
- W. E. Snyder, *Industrial Robots: Computer Interfacing and Control*, Prentice-Hall, Englewood Cliffs, NJ, 1985.
- D. J. Todd, *Walking Machines—An Introduction to Legged Robots*, Chapman & Hall, New York, 1985.
- J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley Publishing Company, Reading, MA, 1974.
- M. W. Thring, *Robots and Telechairs*, Halstead Press, a Division of John Wiley & Sons, New York, 1983.
- and Automation, 1984, 1985, 1986, IEEE Computer Society Press, 1730 Massachusetts Avenue, N.W., Washington, DC, 20036-1903.
- Proceedings of the First International Symposium on Robotics Research*, J. M. Brady and R. Paul (eds.), MIT Press, Cambridge, MA, 1984.
- Proceedings of the Second International Symposium on Robotics Research*, H. Hanafusa and H. Inoue (eds.), Kyoto, Japan, 1985, MIT Press, Cambridge, MA.
- Proceedings of the Annual IEEE Conferences on Computer Vision and Pattern Recognition*, IEEE Computer Society Press, 1730 Massachusetts Avenue, N.W., Washington, DC, 20036-1903.
- Proceedings of the First World Conference on Robotics Research*, Bethlehem, PA, 1984, Robotics International of SME, One SME Drive, Dearborn, MI.
- Proceedings of the Second World Conference on Robotics Research*, Scottsdale, AZ, August 1986, Robotics International of SME, One SME Drive, Dearborn, MI.
- Digital Image Proceedings and Analysis*, Vols. 1 & 2, R. Chellappa and A. W. Sawchuck, IEEE Computer Society Press, 1730 Massachusetts Avenue, N.W., Washington, DC, 20036-1903, 1985.

#### Journals

- IEEE Journal of Robotics and Automation*, IEEE, 445 Hoes Lane, Piscataway, NJ, 08854-4150.
- International Journal of Robotics Research*, MIT Press, 28 Carleton Street, Cambridge, MA, 02142.
- International Journal of Robotics and Computer Integrated Manufacturing*, Pergamon Press, Fairview Park, Elmsford, NY, 10523.
- Robotics Today*, SME, One SME Drive, Dearborn, MI, 48128.
- Robotics World*, Communication Channels, 6255 Barfield Road, Atlanta, GA, 30328.
- The Industrial Robot*, IFS Publications, 35–39 High Street, Kempston, Bedford MK42 7BT, UK.
- Sensor Review*, IFS Publications, 35–39 High Street, Kempston, Bedford MK42 7BT, UK.
- International Journal of Advanced Manufacturing Technology*, IFS Publications, 35–39 High Street, Kempston, Bedford MK42 7BT, UK.
- International Journal of Robotics*, MK-Ferguson Company, One Erieview Plaza, Cleveland, OH, 44114.
- Journal of Robotic Systems*, John Wiley & Sons, 605 Third Avenue, New York, NY 10158.

D. NITZAN  
SRI International

## ROBOTS, ANTHROPOMORPHIC

Anthropomorphic robots, do the jobs of humans, In factories around the world robots perform repetitive tasks once done by man such as painting, welding, and assembly. As robot capabilities advance, anthropomorphic robots will perform tasks requiring higher human faculties, such as adaptability, knowledge, and judgment. In order to behave with greater intelligence, robots will require contributions from the field of artificial intelligence. Although most of today's robots would not be included in a discussion of machine intelligence, the fields of robotics and AI are linked by their history and by current research activities (1). The following discussion defines anthropomorphic robots and presents an overview of con-

#### Conference Proceedings

- Proceedings of the International Symposium on Industrial Robots (ISIR)*, Nos. 1–15, 1970–1985, Robotics International of the Society of Manufacturing Engineers, One SME Drive, Dearborn, MI.
- Proceedings of the Annual Robots Conferences*, Nos. 1–10, 1976–1986, Robotics International of the Society of Manufacturing Engineers, One SME Drive, Dearborn, MI.
- Proceedings of the Annual IEEE International Conference on Robotics*



temporary robots. Research directions are summarized, and likely future applications are suggested.

Early roboticists within the AI community used robots in the development of decision-making (see Decision theory), vision (qv, see also Pattern recognition), and navigation systems. Although primitive industrial robot arms were developed as early as the 1950s, most of the development of intelligent robots occurred in AI laboratories at research institutions such as Stanford (2), MIT, SRI, and JPL (3). Stanford Research Institute's (SRI) *Shakey*, e.g., was built in the late 1960s (4). Acknowledged as one of the first intelligent robot systems, *Shakey* used bumper sensors and a vision system to sense its environment. Its mainframe computer controller used data from the sensors to navigate *Shakey* among obstacles and toward goals. It had no arms but manipulated its environment by pushing boxes with its bumpers. *Shakey* was noted for the ability to do problem solving (qv) in its very limited world (5). Traditional AI disciplines such as knowledge representation (qv), scene analysis, and natural-language understanding (qv) are needed today by robotics researchers to make robots more responsive to their environment and to their human partners.

The most commonly accepted definition of robot in the United States is that advanced by the Robotic Industries Association (RIA):

*A robot is a reprogrammable, multifunctional manipulator (or device) designed to move material, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks.*

The Japanese counterpart (JIRA) of RIA uses a similar, but more liberal, definition that includes nonprogrammable teleoperated devices. The French also use a liberal definition of robot. Consensus has yet to be reached on the exact definition of a robot.

A recurring problem with the definition of robots is that they are defined by what they can presently do but not by what they are likely to be able to do in the near future. The commonly accepted meaning of robot will evolve just as the meaning of computer is changing in response to the ubiquity of word processors and microcomputers. There seems to be no good reason for anthropomorphizing the appearance of robots. It is, however, safe to predict that robots will evolve along the time-tested path of increasing intelligence, adaptability, dexterity, and mobility. A definition of anthropomorphic robots must include reference to the levels of intelligence and adaptability that are now appearing in research robots.

It is illustrative to compare the levels of intelligence in robots to the various levels of intelligence found in animals. Some robots remind one of the small-brained early dinosaurs who were undoubtedly strong but acted slowly and instinctively. Consider instinct to be a form of highly deterministic preprogramming. Other robots may call to mind the domesticated horse, highly intelligent in its ability to learn to adapt to various roles. Perhaps a house-pet robot could be instinctive as a cat or emotive as a dog. Increased communication between robots may lead to the development of a social cooperation reminiscent of a colony of ants or a pack of wolves. New combinations of intelligence, strength, mobility, and dexterity will create robots that defy comparison to the natural world; consider a sperm whale with powerful arms and dexterous hands.

## Contemporary Robots

In a field that is growing as rapidly as robotics, description of the configurations and applications of intelligent robots presents a moving target. Whatever is described today will be passe tomorrow. It is, however, safe to assume that many of tomorrow's robots will evolve from the configurations and applications found in today's factories; and that this evolution will follow paths already determined by current work in research laboratories. Through an understanding of contemporary robots and current research, one can suggest the likely configurations and applications of intelligent robots in the future.

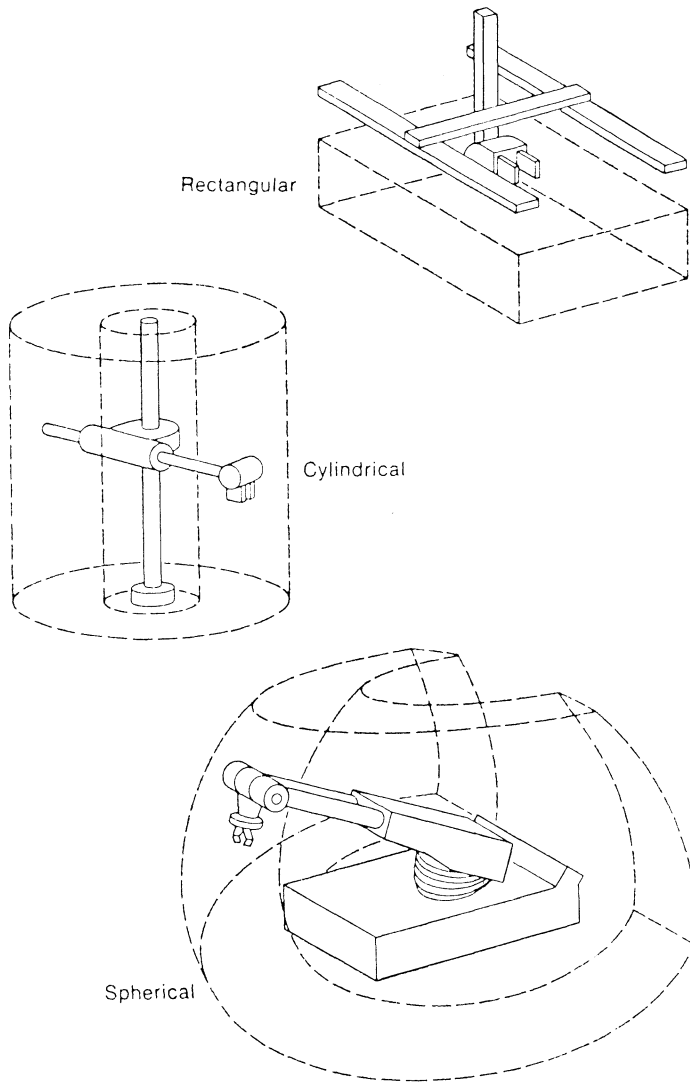
**A Robot Taxonomy.** A robot's capabilities and limitations are determined by its construction, control, and programming. Industrial robots now have only a narrow range of configurations, sensor types, communication capability, and methods of programming; tomorrow's robots will evolve from patterns seen today.

**Parts of an Industrial Robot.** Industrial robots (IRs) are the familiar robot arms that are seen in factories doing repetitive jobs such as welding, painting, assembly, and feeding automatic machines (see also Computer-integrated manufacturing). Each arm is actually part of a robot system composed of three subsystems: the mechanical manipulator (see Manipulators), the actuation mechanism, and the controller. The mechanical manipulator includes the structure and configuration of the physical arm itself. The actuation mechanism refers to the type of power and method of moving the arm. The controller is the computer that controls the movement of the arm. An effector, or end effector, is the hand or tool mounted at the business end of the arm. Since the effector may be custom built or changed frequently, it is not considered to be a main part of the general-purpose IR. The combination of the above components defines the strength, precision, dexterity, and programmability of the robot (6).

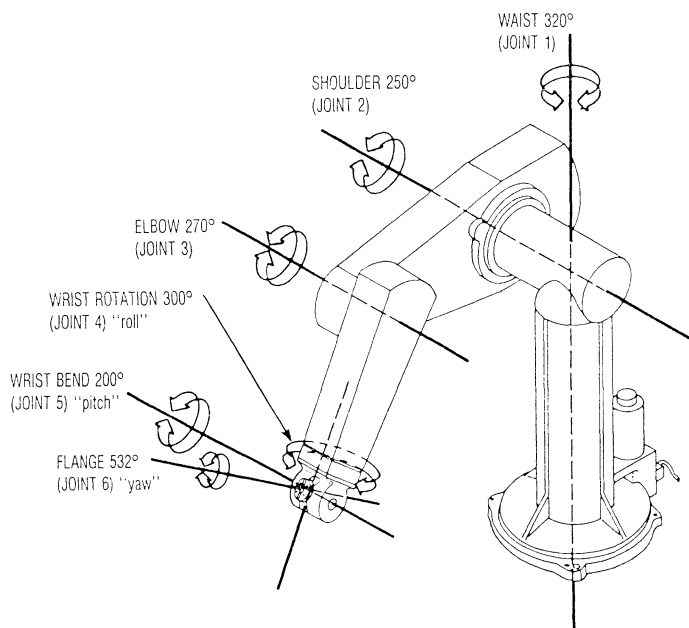
The mechanical manipulator of an IR is generally constructed of rigid metallic members configured into axes defined by a series of joints. The joints, usually from three to six in number, permit the members to rotate or slide with respect to adjacent members. Common practice has dictated that the first three axes determine the work envelope of the robot (Fig. 1), and the remaining joints determine the flexibility of the wrist in its ability to orient the hand or tool (Fig. 2). The axes around which a robot can move are termed its degrees of freedom.

The work envelope is the physical volume in space that can be reached by the end of a robot arm, not including the hand or tool. Figure 1 depicts three basic IR configurations in use today. The SCARA (Selective Compliance Assembly Robot Arm), not shown, is an increasingly common configuration developed by the Japanese. Robots gain additional degrees of freedom when they are mounted on a track, X-Y table, or mobile platform. Conversely, IRs with more limited requirements may have fewer degrees of freedom than the common six-axis robot.

The end effector is often supplied as a gripper with two parallel jaws that may or may not be servocontrolled. A non-servoed hand only opens or closes, whereas a servoed hand may open or close an arbitrary amount. Although intended as a general-purpose hand, parallel jaws cannot reliably grasp many irregularly shaped parts. Parts that have been grasped



**Figure 1.** The work envelope of a robot is determined by its configuration. (Source: IEEE).



**Figure 2.** Degrees of freedom of a typical robot arm. (Source: Unimation, Inc.).

may tend to slip from a general-purpose gripper, and delicate parts may be damaged by the fingers. Special-purpose effectors or tools solve this problem for robots with fixed applications such as spot welding, but custom-made effectors are very expensive.

The actuation mechanism that powers the IR has traditionally been hydraulic, pneumatic, electric, or some combination of these. The mechanism may include the important ability to provide feedback control (see Cybernetics) to correct mechanical positioning through the use of servomechanisms. Hydraulically powered robots tend to be very strong but are also dirty and noisy. Pneumatic robots tend to be mechanically simple and lightweight but noisy due to their air compressors. Most modern IRs are electrically powered, giving them cleanliness, quietness, moderate strength, and easy availability of power.

Feedback is provided by position resolvers located at each joint that enable the robot system to know the absolute position of each axis to within a few thousandths of an inch regardless of the unavoidable play and wear in the mechanical linkage. This feedback system is important in the understanding of the robot system as a fundamental example of closed-loop adaptive control. Closed-loop adaptive control has a strong impact on the ultimate intelligence of a robot.

The controller in a modern IR is a computer that stores the robot's program and controls the robot's motions by communicating with the actuation mechanism. (Robots using controllers with mechanical stops or hardwired plug-in boards are inexpensive and efficient for factory applications such as material handling; however, they are not discussed here because they are so much less adaptable than computer-controlled, servocontrolled robots.) The controller stores the points that make up the robot's program and can often generate signals or await signals from external sensors at designated times during the program.

**Sensors.** Consider a strong and dexterous workman on the shop floor with the ability to perform repetitive tasks accurately and tirelessly. Further consider the workman to be completely blind with no sense of touch. Most IRs, like this workman, are without the benefit of sensory input. Sensors (qv) gather data for adaptive control systems, e.g., to supply guidance information for robot movements or to provide measurements for quality assurance and inspection systems. Sensors provide IRs with vision, touch, and other senses, enabling them to explore and analyze their environment in order to behave more intelligently.

Several types of sensors are currently in use to provide the IR controller with data of varying complexity, from the simple on-off of a light-emitting diode (LED) sensor, through the scalar output of a strain gauge, to the complex data generated by a vision sensor. Most IR sensors can only generate the limited information of binary on-off states. In contrast, advanced vision systems now can analyze their sensory input to supply data on part identification, orientation, and measurement. Recently, tactile, acoustic, and laser sensors have been used to measure force and shape, provide range data, and analyze the quality of welding processes. Sensor technology is one avenue by which the intelligence of robots may be greatly increased. It is a very active field of research (see Multisensor integration).

**Communication with Other Computers/Machines.** Industrial robots seldom work by themselves. They need to communicate to the automated machine tools, material-handling systems,

inspection stations, and other automated devices that comprise the robot's work cell. If the IR needs more than one program, it may be downloaded from a larger computer. Internal communications within the IR controller are required for the hierarchical control of the multiple processors used in some IRs.

Up to now these communication links have provided very limited capabilities. Robot communication with other machines has traditionally been handled by binary-valued digital input-output (DI/DO). This limits the information content of the communicated data to yes-no values unless the user programs the controller to read several DI/DOs as one signal. In that way eight DI/DO ports could be combined to provide one 8-bit port capable of values 0-256, still a limited amount of information requiring the dedication of several precious ports. Complex data from a vision or taction system cannot even enter such a controller.

Communication with other computers has been provided through the RS-232 ports that are standard with most contemporary robot controllers. Although RS-232 has been adopted by the computer world, it is really not suited to handling high volumes of complex or real-time data or to a networked environment. Communicated data must be interpreted by the IR controller or its program, but no standards currently exist for the representation of anything of value to the IR other than complete programs and program point files. Information on part geometry or characteristics, e.g., is not represented in any consistent way in manufacturing system databases.

Communication for hierarchical control within the controller has been implemented on some robots, such as the Westinghouse-Unimation PUMA. A minicomputer functioning as the robot controller supervises a series of microprocessors that function as joint controllers. Much of the number-crunching load of angular computation, motor commands, and feedback adjustment is thus removed from the main processor. This system allows much faster control and allows the supervisory controller to assume the higher decision-making functions of robot intelligence.

**Methods of Programming.** Much of a robot's intelligence is determined by the extent to which the knowledge and/or skill of a skilled workman can be imparted to the robot. This is accomplished by various methods of programming the robot. These methods range from the use of simple mechanical stops, through teaching the robot by physically or graphically moving it through its program points, to the use of special robot-control languages. Each method has advantages and limitations, especially as it relates to adaptability and ease of programming.

The oldest and simplest method of robot programming involves the setting of mechanical stops that limit the travel in each direction of selected axes. Skilled machine operators easily learn new mechanical skills, and they intuitively understand this method of programming. Although even modern SCARA robots may use mechanical stops on one of their axes, their great disadvantage is that the program cannot be modified under computer control.

Most IRs in today's factories are programmed by physically teaching the robot each point in its program by either the lead-through or the teach-box method. Applications such as spray painting require that the robot's paint gun follow an irregular, but continuous, path defined by thousands of points. In the lead-through method an expert painter physically guides a counterbalanced arm through the painting motions. The con-

troller samples the position of each joint several times a second, recording them as the robot program. This method permits the robot to paint as well as the best painter on his best day, i.e., if his best day happened to be when he programmed the robot. Programs composed of so many points are impossible to edit, so that major segments of the program must be retaught each time the program is modified. The lead-through method does not lend itself to decision-making capabilities within the program.

More IRs are programmed with the aid of a teach box than with any other method. The programmer instructs the robot to move from one critical point in the program to another by pressing buttons or switches on the teach box or teach pendant. At certain points the operator presses a button on the pendant to record the current joint positions in the controller's memory. The program thus consists of a number of individually taught points the robot replays each time the program executes. Some teach pendants permit the inclusion of branches or canned subroutines for the performance of common functions such as calculating the position offsets in a pallet matrix. Teach-box programming attempts to bridge the gap between mechanical programming and language programming so that a skilled workman rather than a computer expert may still program the robot.

Graphical robot-programming systems, such as McDonnell Douglas Automation's PLACE System, involve a newer concept in which the robot is programmed off-line by moving a computer-generated graphic image of the robot to the various points in its program. Relevant objects in the robot's environment are also modeled on the graphics screen. A teach box or graphics tablet may be used to move the image. The programmer performs preliminary testing of the robot program using the graphics terminal. He visually determines whether the robot is correctly performing the task. Important debugging steps such as collision detection can be attempted visually, but this method is prone to human error. Off-line graphical programming is desirable for planning and preliminary testing of a new cell layouts and robot programs without taking the existing cell or robot out of production. In reality, fine tuning of the program must be performed on the physical robot in its actual environment because the computer models of the robot and cell are idealized, and most simulate neither normal variance from mean specifications nor arm mechanics such as acceleration and behavior under load.

The use of language has long been regarded as a hallmark of intelligence. Specialized robot-control languages are so recent that one of the first such languages, VAL from Westinghouse Unimation, is still in common use (7). Early languages, termed first-generation manipulator languages, were similar to BASIC with special commands added for robot motion and sensor I/O. A programmer enters the program by typing the steps on a keyboard. The program can be listed on the terminal screen, and commands to save or load a given program can be given through a rudimentary operating system. Although the teach pendant may still be used for the definition of specific points, the points are named using the keyboard and are referenced in the program by their names. Otherwise-skilled workmen require computer training before they can use a language to program the robot. Since programming with a special computer language, as well as simply typing on a keyboard, is often totally outside their experience and self-image, much resistance to this type of programming has arisen from the factory floor. The first-generation lan-

languages are also inadequate from another standpoint. They only begin to tap the power traditionally available with a full-featured computer-science language and operating system. Programmers and manufacturing engineers could envision advanced applications that the limited languages could not support.

Second-generation programming languages are designed to remedy the inadequacies of the early languages. These languages are based on full-featured, structured computer science languages; e.g., the RAIL language developed by Automatix is based on Pascal, whereas IBM's AML (A Manufacturing Language) is a completely new structured, interpreted, language. They offer additional mathematical operations on program points, a variety of data and control structures, and improved communication with sensors and external computers or machines. Since the new languages are procedural, they can be used to build subroutine libraries and application packages.

Application packages, such as spot welding or assembly packages, are expected to provide the missing link between the advanced programming skills required to create the application packages and the production expertise required to apply them. Application packages and toolkit libraries may stratify robot programmers into a three- or four-level hierarchy. In a software development firm one may find systems programmers creating the subroutine tools. Application programmers will use these tools to create easy-to-use application programs. In the factory a manufacturing engineer will install the software and will be responsible for physical and logical communications between the robot and other computers and devices. The robot operator will use the installed program to program the robot to handle each of its assigned tasks. In this way the people with the production expertise are shielded from the complexity of the programming language, yet each person is provided with an avenue for the input of his expertise. This type of programming hierarchy is responsible for the evolution of office automation through application packages such as WordStar, dBase III, and Lotus 1-2-3.

**Applications.** About half of the IRs in factories around the world are working in welding or material-handling applications. The next most common applications are assembly and painting/finishing (8). Some of these tasks are similar to the simple, repetitive tasks first performed by IRs; others are on the cutting edge of applied technology. It is helpful to divide discussion of IR applications into those in common practice and those regarded as state of the art. The former category is more frequently encountered, but the latter is more useful in a discussion of intelligent robots. Exploring the state of the art also helps to determine the direction of robot evolution.

Over one-third of the IRs in the United States perform welding tasks, primarily spot welding. Common practice finds these robots programmed by a teach pendant to move to a point, squeeze and weld with the spot-welding tool, open the tool, move to the next point, weld, etc. There is limited use of sensors, perhaps no more than a signal that an auto frame is in place and ready. The robots are taught as many welding programs as there are kinds of auto bodies on that production line. The required programs are downloaded from a computer outside the robot's controller to each of the robots in the welding line (each has its own distinct program). These robots cannot inspect their own work, nor can they identify what part they are welding. They work more rapidly and consistently than human spot welders, but they still cannot weld in some

awkward positions. Humans do the trickier spot welding. This is the classic dumb industrial robot reminiscent of early spray-painting robots that sprayed workers, equipment, missing parts, and anything else that got in their way.

Most material-handling applications are of the pick-and-place variety, which the simple nonservocontrolled robots often can do well. The IR may pick up a part from a feeder fixture, perhaps after receiving a signal from an LED sensor that the part is present and in the correct position, and place it in a pallet. The IR may load/unload parts to/from a pallet by using palletizing routines that are most likely supplied with the robot-control software. Through a simple language like VAL, or through a complex teach pendant, the programmer teaches the IR the exact location of the first cell in the pallet and the number and dimensions of cells in each direction, and the palletizing routine handles the rest. Material-handling IRs that retrieve parts from conveyors may be constrained by a need to have the part stationary and in a perfectly registered location and orientation.

Machine loading/unloading applications are related to material-handling applications. An IR performing machine loading/unloading is centrally located in a work cell such that each machine it tends is within its work envelope. A machine-tending IR may get a part from a receiving pallet, load it into a CNC (Computer Numerical Control) machining station, unload the part when it is finished at that station, present the part to an inspection station, retrieve the part from the inspection station, and place the part in either an output pallet or a reject bin (depending on the signal from the inspection station). Machine loading/unloading applications require an IR with the capability to send and receive signals to a variety of devices. The communication requirements of such applications require the capabilities of a robot-programming language and the services of a trained robot programmer.

State-of-the-art applications require more sensory and decision-making capability on the part of the IR system. Assembly and arc welding are two such applications. In an assembly operation the IR handles several different parts, each of which must be placed, inserted, screwed, or otherwise manipulated with high precision. Assembly applications may require the use of sensors such as the LED and force sensors in the fingers of the IBM 7565 robot system. Advanced use of sensors in the robot system lessens the requirement for expensive fixturing and off-robot sensing and communications. The IR can determine by itself whether it has correctly gripped the part or whether an insertion has been completed or is jammed.

Some state-of-the-art IRs use vision systems to further reduce custom fixturing and sensing. Vision systems help the robot to locate a part, to identify the part, and to determine its orientation and how to grip it. Some systems compute the location of moving parts such that the robot can pick up the part from a moving conveyor. Vision systems enable the IR to adapt to partially filled pallets and randomly oriented parts in material-handling and assembly applications. Limited inspection can be performed by commercial vision systems.

Highly specialized vision sensors are required for some arc-welding applications, such as butt joins or corner joins of flame-cut plate. Arc welding of flame-cut plate is difficult for a robot because the algorithm of perfect repetition breaks down. In fact, one of the strengths of the arc-welding process is that it can accommodate the nonuniform edges produced by flame cutting. A human welder uses his eyes to follow the actual welding path, which deviates from an ideal path that can be

geometrically described for all such parts. The human knows that the moving welding torch welds quickly where the gap is narrow but that it must slow down where the gap widens to allow more metal to flow. In order to complete the weld in a single pass, an arc-welding IR requires real-time sensing to control the path and speed of the welding torch. Special-purpose vision sensors are interfaced to an advanced control language to provide this real-time sensing and control. A few such systems are now out of the laboratory; some experimental systems have used laser sensors to enhance the quality of the completed weld.

Nonindustrial applications of robots have made headlines for their contributions to space exploration, undersea salvage, nuclear inspection and salvage, bomb disposal, aid to the handicapped, and special-purpose military systems. These applications are not now in the mainstream of robotics, although the definition of mainstream robotics will change with time. Some of these applications are excluded from the present discussion of intelligent robots for a variety of reasons: the man-in-the-loop of the space shuttle robot arm, the earth command of *Voyager* and *Viking*, the telechiric control of undersea, nuclear, and bomb-disposal robots (9), and the highly specialized robotic military devices such as the cruise missile. Robots designed to aid the handicapped are still in the research stage, primarily using time-tested industrial methods combined with speech technology. Some nonindustrial applications that are expected eventually to make strong contributions are natural-language processing robots for the handicapped (see Protheses) and highly adaptive mobile robots designed for a battlefield or a planetary surface.

### Current Research

The capabilities of tomorrow's robots are being created today in academic and industrial research laboratories around the world. Advances in hardware and software will endow future robots with greater autonomy and adaptability by providing them with improved senses of vision and touch as well as an enhanced capability to navigate and otherwise plan actions in less deterministic environments. Progress in control systems, communications, and methods of programming will also enhance robot capabilities.

**Control Systems.** Some control research is intimately tied to the development of new robot hardware. For example, research in compliance attempts to equip the robot with hardware (under software control, of course) that is capable of dealing with environmental uncertainty. This ability of the robot to react to environmental forces is termed compliance (10). A more unusual control problem is that of controlling legged locomotion. Researchers at CMU (11,12) have developed working models of one-legged, two-legged, and four-legged robots that hop, run, or trot.

Each new capability of IRs adds to the already heavy computational load on the IR controller. There are two solutions to this problem. This first involves the development of faster and more powerful processors to a point at which a single processor can perform the multitasking and real-time operations required of an intelligent robot. The second solution involves the use of multiple processors functioning together. It is difficult to program a single-processor controller to simultaneously perform high-level path planning, calculate joint positions and act on resolver feedback, accept and interpret sensory data,

and send and receive signals and data. Researchers at the Automated Manufacturing Research Facility (AMRF) of the National Bureau of Standards (NBS) are continuing to develop a hierarchical control and feedback model for the coordination of multiple robot processors (13). This system can be compared to a human hierarchy that uses management by exception. The lower level processors are assigned tasks by the upper level processors. The lower levels go about their tasks without bothering the upper levels until they are finished or until an ambiguity or exception occurs. The upper levels assign tasks, make decisions based on reports from below, and coordinate the multiple lower processors. This is one promising method of dealing with the overwhelming amount of sensory, feedback, and control data with which tomorrow's robots must deal.

The opposite problem occurs when a single higher level entity must coordinate a robot with multiple arms. There are many operations that two arms could perform without the custom fixturing necessary today. Consider the actions performed by humans with two arms, e.g., opening a jar with a screw-on lid. It is necessary to monitor and control the applied forces from both arms while planning, controlling, and monitoring each arm's path. Tightly coordinated coprocessing is seen as necessary for successful multiple-arm implementations, but it is truly difficult for the robotic right hand to know what the robotic left hand is doing. Consider the human brain and its coordination of the human's two arms. The coordination data passes between the left and right hemispheres through the corpus callosum, the thickest bundle of nerves in the human body. Just as a massive amount of communication is suggested by the human architecture of coordination, the various components of a multiarm robot system will most likely have massive communication requirements. The problems in multiple-arm control are also found in the control of humanlike fingered "hands" and in the control of multiple devices in a manufacturing system. Early research in multiple-arm coordination was carried out at Stanford and is continuing at MIT, Lehigh University, Westinghouse, and elsewhere (14,15). *Odex 1*, a six-legged mobile robot developed by Odetics, Inc., provides an impressive example of the coordination of six independent manipulators. *Odex* is rather slow due to the communication and computation requirements of its distributed coprocessing architecture.

Related to hierarchical control is the problem of distributed processing. Distributed processing is desirable when several robots are performing coordinated processes within a larger computerized entity such as a flexible manufacturing system (FMS). When various levels of decision making are performed in different computers, sometimes even at different locations, the robot system is much more difficult to program, test, and maintain compared to the single-robot, single-controller model. Not only must a systems approach be used but also hurdles of heterogeneous equipment and near real-time response through networks must be overcome. Research in distributed processing is being conducted primarily in the computer field rather than the robotics field.

A smarter IR will need access to a knowledge base, a database of stored part and process data that the robot controller can use to modify its own program variables and data. The database may be similar to today's CAD (Computer-Aided Design) database in which part geometries are described. The largest problem is that the kind of information the robot may require is not currently indexable in CAD databases. For example, if a vision sensor reports that it sees a cylindrical part

of certain dimensions, with a keyway visible on one end and a taper on the other, there is currently no standard indexing or searching scheme through which the part descriptions could be searched in the CAD database to identify the part. Current standards do not support data communication between a robot controller and a CAD database. The real problem is that of describing and interpreting shape, material, and process data. For example, the flexible representation of shape is a major unsolved problem (16). The shape-description systems built for graphics and CAD systems do not approach the powerful natural shape-representation systems used by humans and animals.

Knowledge-base research is pursuing two avenues: proprietary systems developed by vendors and generalized systems developed through group technology (GT). Some CAD system vendors are creating proprietary links, with proprietary software, between certain robots and the vendor's CAD databases. This is the fastest way to progress, but it forces the manufacturer to limit himself to one vendor, which may cause expansion or compatibility problems. Group technology refers to a system of classifying parts or processes by their relevant attributes such that parts related by their geometry, material, or process may be retrieved from the database. Both the generalized system of classification and the system of retrieval are occupying researchers, e.g., at Arizona State University.

As sensors improve, more and more reliance will be placed on sensory data; less reliance will be placed on expensive fixturing. Eventually, sensor systems may direct most of the operations of an automated machine or robot. One example of sensor-controlled automation (SCA) is that of a robot arm that navigates by a vision sensor on its hand or wrist. The sensory data and the robot controller continuously interact, mimicking human eye-hand coordination. Advances in sensory data throughput and interpretation are required for SCA to become practical. It is likely that automated machines with more constrained environments will be the first implementations of SCA.

Expert systems (qv) are traditionally regarded as inference engines acting on rules that capture the basis of a human expert's decisions. An example of an expert system's contribution to manufacturing may be found in tool selection. Based on material, dimension, process, and cutting speed, an expert system could use its rules to select the appropriate cutting tool for the job. Current expert systems function best in extremely limited domains. For example, an expert robot system has been developed for welding applications (17). It is likely that expert systems will be so thoroughly integrated into the control software of robots that the term expert system may not even be used. Instead, the robot will simply be expected to behave more intelligently through rule-based behavior. As currently conceived, robot expert systems will not increase the general intelligence or common sense of the system. The robot will be more intelligent only in a narrow area such as the interpretation of complex sensory data from vision or taction sensors. If the robot's interpretation is a member of the set of normal interpretations, the robot will act; otherwise, the system may ask for mediation from a higher level entity, be it machine or human. A good robotic expert system would be able to handle some of the ambiguities of the real world, responding to probability thresholds in the absence of certainty. The expert system would also be able to aid in the debugging of the robot's programs by explaining the chain of reasoning behind its actions. Up to now expert-system techniques have

been query-oriented; a much more comprehensive approach must be taken to develop autonomous decision-making ability.

Autonomous decision making will first be implemented in very limited domains whose parameters are completely known or can be determined by the robot. Winograd's *blocks world* (see SHRDLU), e.g., has been physically implemented with robots at several universities. In this model the robot system knows the location and characteristics of every object and action that can take place in its environment. This information makes up a dynamic knowledge base the system must continually update with data on locations, speeds, a history of actions (should the system need to backtrack), the task at hand, and the current, past, and future subtasks. The system handles complex tasks by decomposing them into simple steps that it knows how to do.

This type of limited-task domain is quite appropriate for a highly constrained manufacturing workcell. The problem becomes vastly more complex when the robot's environment is less constrained. Work is being done on this problem at CMU and NASA. The CMU Terregator is designed to autonomously navigate outdoors along ordinary paths or walkways. NASA is working on a mobile robot to explore the surface of Mars. Since the communication float between planets causes a delay with each transmission that requires a response, it is desirable to have as many decisions as possible made autonomously on Mars. Earth decisions will be requested when the robot cannot interpret ambiguous or conflicting information with a high degree of confidence.

IRs currently operate in a highly constrained domain. Inexpensive, simple, reliable IRs can be used in such applications, but the cost of the physical robot accounts for only one-third of the total cost of the robot system. Programming accounts for one third, and fixturing accounts for the remaining third. Increasing the autonomy of the robot greatly increases its cost but promises to drastically reduce fixturing costs and potentially reduce programming costs. Whether autonomous IRs become commonplace may depend more on their economics than on their technological feasibility.

**Methods of Programming.** Programming presently accounts for one-third of the cost of an IR system. This represents the costs of vendor-supplied software, highly trained programmers, and downtime of the IR during programming, debugging, and testing. Reprogramming for flexibility and process change implies a high continuing cost of system software maintenance. One programming trend lies in the use of general-purpose computer-science languages that may be extended to include robot-motion commands. These general-purpose languages may be used by programmers with existing expertise to create process applications packages that are easily used by manufacturing personnel. Languages such as LISP, Pascal, and C are now commonly used in robotics-research laboratories (18). FORTH, an excellent language for real-time control, is used to program the robotic arm on the space shuttle. Ada, the language developed by the Department of Defense, has been experimentally extended to provide packets of robot-control commands (19).

Several methods of programming are emerging that address other parts of this problem. Research is improving the user interface to the robot system so that the robot may be programmed or reprogrammed by manufacturing personnel without a high level of computer programming expertise. One obvious way to do this is to replace the present interface, a



highly technical robot-programming language, with a front end that uses common English to be typed at a keyboard or perhaps spoken to the robot. Menu-driven programming systems are now under development and testing.

Another way to ease the programming problem is through the development of task-level or object-level languages (20) through which the user would specify the task to be performed (or the processes to be performed on the object) in much the same manner as is now used in the common written process sheet. Researchers at Stanford have made progress on a prototype task-level language called AL (21), and IBM has created the specification for a task-level language called AUTOPASS (22). At the University of Edinburgh the APT language used to program NC machine tools has been extended to include robot-control commands and some task-level capability to describe assemblies (23). The language, called RAPT (Robot APT), has shown initial success, but it may not accommodate the level of real-time adaptive response to sensor input that is required of a true task-level language.

A more visionary and far-reaching approach is to develop a robot that learns on the job. Such an IR would then learn to deal with anomalies as they come up, generalizing them after enough experience, and eventually deducing from the generalities reasonable solutions to new situations. Such robots may use a learning program similar to EURISKO (qv) (24), the most successful experiment in machine learning to date. Although EURISKO has discovered new knowledge in several domains, its slow speed and the low ratio of its productive to unproductive solutions may restrict its value to robotics. As EURISKO-like programs improve, they may significantly revise our notion of intelligent robots. The success of robots that learn will come first in highly constrained domains (see Learning) (25).

**Sensors.** One of the most active areas of research today aims to improve the sensors that give a robot the senses of vision and taction, to improve the internal representation of objects that the sensors detect, and to integrate (or interface) the ability to process complex sensory data within robot-programming languages. Some robots developed to aid the handicapped accept spoken commands through a primitive sense of hearing, but speech recognition (qv) has not been focused on industrial applications. Although there has been no emphasis on the development of the other human senses of smell and taste, some senses that have no human analog are showing promise in the laboratory through the use of laser, ultrasonic, and esoteric sensors. Vision systems composed of sensors and software to manipulate and interpret visual images are a key to greater IR adaptability and flexibility.

Vision systems are necessary for a variety of tasks, listed here in the order of increasing complexity (26):

- identification/verification of objects and their current stable state;
- location of objects and determination their orientation;
- visual inspection for incomplete or defective parts (cracks, etc.);
- visual servocontrol (arm guidance);
- scene analysis and navigation for mobile robots; and
- complex inspection of parts and assemblies.

Vision systems now available in the commercial market can perform subsets of the first three tasks. Commercial vision

systems now use binary images to extract basic feature sets from part images. These systems are useful in highly constrained manufacturing applications, but they have limitations when parts have many stable states, when there are different kinds of similarly shaped and sized parts, and when parts may be overlapping as in the bin-picking problem. Binary-vision systems function by reducing the information in a scene to either black or white, but in the process the potential richness of the visual data is lost. Systems now in research laboratories promise more generalizable approaches that lend themselves to more intelligent applications.

Vision researchers are studying several areas:

- use of gray-scale (instead of binary) images to allow more detail;
- improved algorithms and parallel processing to allow gray-scale image processing in real time;
- use of VLSI chips to produce low cost, high-resolution intelligent camera systems;
- various 3-D vision approaches; and
- systems for representing or reasoning about visual data.

Several of these research topics have been combined to allow, e.g., practical gray-scale image analysis (27). Optimized algorithms and parallel processors have been put on VLSI chips at MIT, Hughes, and Westinghouse. New vision systems handle image processing and basic analysis, passing to the robot only final data on part identification, location, and orientation. In this way the robot language and the controller need not be burdened with computation-intensive image processing.

Visual servocontrol, or guidance, of robot arms is comparable to the visual guidance of human arms. The comparison of a blind person to a sighted person is appropriate. Both a blind person and an IR use memory instead of senses to determine the location of objects. Sighted individuals need not have everything in their environment prerecorded in memory. Visual servocontrol allows a robot to find an object that it wants and to maneuver with respect to the object in a desired manner for inspection or manipulation purposes. NBS has developed a vision and control system for this purpose.

Three-dimensional vision systems are needed when the objects to be imaged do not lie in a predetermined plane, such as a conveyor belt, or when 3-D data is needed to identify the object or its stable state. Several techniques can be used to create 3-D imaging systems. Human binocular vision can be emulated by stereo vision (qv) systems. These two-camera systems are the most common 3-D approach in the laboratory, and some systems are commercially available. Structured light and triangulation, developed at NBS and elsewhere, involves rigid constraint of the lighting. The data reduction achieved through the use of structured light allows more rapid real-time image analysis. General Motor's CONSIGHT system has successfully used a modification of the structured light technique by combining structured lighting with a line-scan camera. Laser and ultrasonic range-finding devices have been investigated more recently.

Vision research includes the interpretation of the image and reasoning about its relationships. These problems, listed below, are more difficult to solve than the issues discussed above because they deal with what may be described as the higher level thinking skills of analysis, synthesis, and evaluation:

- representing knowledge about objects, particularly geometry and spatial relationships;
- developing methods for reasoning about spatial relationships among objects;
- understanding the interaction between low-level information, high-level knowledge, and expectations;
- interpreting stereo images for range and motion applications; and
- understanding the interaction between an image and other information about the scene such as written descriptions.

Stanford's ACRONYM vision-interpretation system is one such model-based vision system that addresses some of these questions (28). It is difficult to predict specific results from the research in vision systems because of its dependence on advances in VLSI and in knowledge representation.

Tactile sensing is another active area of sensor research, but the state of the art in taction technology is primitive when compared to the advances made in vision systems. Tactile sensing may be defined as the continuous sensing of variable contact forces, typically by a sensor array (29). Researchers are trying to move beyond present taction applications that use simple limit switches or force/torque sensors in the IR fingers, hand, or wrist. There are three major categories of tactile sensor applications:

- simple force sensing and slip sensing;
- object position and orientation determination; and
- object identification.

Each application category requires a different approach to sensor design and control.

The most promising tactile-sensor research follows one of three broad designs: electrooptical, piezoresistive, or piezoelectric. Providing a true sense of touch implies computing the 3-D shape of the object being touched. The first such tactile sensors to do this used passive substrates and conductive elastomers, but they were low-resolution devices prone to hysteresis. High-resolution tactile sensors have been developed at the Jet Propulsion Laboratory at Cal Tech that use an array of force-sensitive elements with solid-state transducers. Microcomputer elements transform the complex sensor data to 3-D data. The addition of such a sensor to a robot finger makes a smart finger that can find edges and follow arbitrary contours. More recently, work has progressed on developing an artificial skin by including thermal sensing in addition to the force sensors to simulate the thermal and tactile sensing properties of human skin (30). Researchers at the University of Pisa have tested the artificial skin both in industrial part identification and in prosthetic devices for the handicapped. Active research is also being conducted at MIT (31) and at the University of Florida (32).

Tactile sensors are being applied to research to develop a compliant three-fingered hand (33) at MIT, Stanford/JPL, and Hitachi. While a compliant hand requires taction sensors in the fingers, a compliant wrist may require additional sensing capability at the wrist joint. The remote center of compliance wrist (RCC), developed at Draper Laboratory, allows passive compliance of the robot wrist during assembly operations such as inserting a snugly fitted peg into a hole. Since the RCC has been commercialized, Draper researchers have instrumented the RCC to provide active feedback to the robot controller. The

instrumented remote center of compliance wrist (IRCC) represents the leading edge of technology in wrist sensors. It allows the robot to follow contours, perform insertions, and incorporate simple touch sensing into robot-control programs.

A variety of special-purpose sensors that have no human analogue are now under development, such as laser and ultrasonic rangefinders. Depth maps can be generated with ultrasound or with parts of the electromagnetic spectrum beyond the limited range of human vision. For instance, infrared sensors can detect humans in the robot's work space.

The continued evolution of the adaptive, closed-loop control that is called intelligent behavior rests not only on advances in sensor technology but also on the interpretation of the complex data that advanced sensors return. Model-based sensor systems are expected to provide future manufacturing sensor systems with complex analysis capabilities by including knowledge about manufacturing processes, CAD, simulation, and control algorithms. The analysis capabilities will surpass the monitoring and control capabilities of human operators by being more sensitive, more precise in analysis, more rapid in feedback response, and more decisive in corrective action. Future IRs will thus work to very fine tolerances while they maintain consistently high quality control (34).

**Planning.** A major component of robot intelligence is the extent to which the robot plans actions before they are executed. To anthropomorphize, an intelligent robot thinks before it acts. The ability to plan may even become a reasonable yardstick by which to measure robot intelligence. Whereas the ability to perform simple path planning (qv) between two points is already built into most robot controllers and languages, advanced path planning includes grip planning, collision avoidance, and navigation (for mobile robots). This kind of planning cannot be done by the programmer because the IR must respond to arbitrary or unexpected changes in its environment. Planning (qv) requires that up-to-date knowledge of the environment be available to the robot for real-time decision making.

**Grip Planning.** Grip planning concerns the automated decision of where to pick up a part and with what grasping force, both to ensure a firm grip and to avoid damage to the part. It affects the path planning of the overall arm because the arm must be positioned such that the wrist is in the orientation determined by the grip-planning algorithm. This is a hard problem even when the 3-D solid model of the part is captured and the geometry of the gripper is known. Consider the difference between the grip necessary to handle an egg-shaped billet of aluminum with that necessary to handle a chicken egg. Grip planning in a loosely constrained environment must be informed not only by a vision or taction system but also by a database that includes knowledge of materials and mechanics.

**Collision Avoidance.** Collision avoidance is often discussed with collision detection, but they are quite different issues (35). Collision avoidance is a lot harder. One only need consider the domain of driver education or that of the air-traffic controller to conclude that, although it is important to detect collisions, the emphasis should be on avoidance. Robot software will simulate the proposed action to determine whether a collision will occur; if a simulated collision is detected, it will plan an alternate path or action. This real-time simulation will occur simultaneously with the execution of the robot's task. It bears an interesting resemblance to the kind of forethought found in humans. Progress has been made in planning

by confining and defining the problem domain. The open-domain approach used by early AI researchers has not been successful in developing practical software that can adapt to anomalies in a domain filled with ambiguity.

**Navigation.** Navigation was an active area of early AI robotic research, but the problems associated with an unconstrained environment frustrated total success. Accurate positioning of robot carts without the use of expensive fixtures still eludes researchers. Such robot carts are ubiquitous in some foreign factories, such as Renault in France and Fanuc in Japan. Their navigation problem is partially solved by a guidance system that uses wires buried in the factory floor. Though successful, the wire guidance system constrains the mobile robot, and the tough issues of navigation are avoided. Similar systems use homing beacons and triangulation methods (36).

There is growing interest from NASA, the nuclear power industry, and the military in the development of robotic navigation in unconstrained environments. NASA, e.g., is designing a mobile Martian explorer that can handle most of its navigation and decision-making tasks onboard. The nuclear industry is interested in robots for the emergency repair of nuclear facilities, and the military is investigating an autonomous land vehicle. Each application has in common the uncontrollable aspect of the environment (37).

Several methods for the navigation of a robot explorer are being investigated. In the configuration-space method obstacle-free zones are identified in the navigation space (38). The bloomed obstacle method involves bounding all obstacles with circles (39) or polyhedra (40). Another method describes global navigation as the connection of a legal highway in one known area with others in other known areas (41). There has been recent interest in combining autonomous navigation with learning capability (42). The new navigation research seems to pick up where *Shakey's* development ended.

Researchers at modern robotics laboratories have combined navigation techniques with advances in vision systems, improved algorithms, and faster and more powerful computers to produce new vehicles such as the CMU *Terregator*, which can slowly navigate along the kinds of paved paths used outdoors by humans. Since the *Terregator* relies on the analysis of massive amounts of visual data, practical applications will have to wait for much faster processing speeds, most likely obtainable through parallel processing. Although the creator of one advanced personal robot claims it can position itself in its home space to within one millimeter of its absolute position through the use of enhanced ultrasonic rangefinders and an internal map of its environment, progress in practical navigation and absolute positioning in unconstrained environments is expected to be slow.

**Communication.** Compare the potential output of a group of independent workers with that of an organized work force. Similarly, single robots, programmed to stand alone, cannot match the synergy and efficiency of an integrated system that includes several robots working with automated machining centers and material-handling systems. This concept is known as computer-integrated manufacturing (CIM) (qv). Computers are used to control and optimize the manufacturing process through an electronic coordination that relies heavily on communication between each of the elements of the CIM system (43). The communications must be two-way, between each machine or robot and a higher level controller, so that programs and specifications from planning software and manufacturing

databases may be passed down the hierarchy, and feedback for dynamic optimization may be passed up. Progress in this area is expected to be rapid as communication systems now under laboratory development and testing are accepted as standards by the manufacturing community. Such standards are required in manufacturing for high speed networks, data structures, and interfaces.

**Networks.** In addition to the complex sensory data discussed above, robots employed in a CIM system will require real-time communications to and from other machines or robots in its work cell and to and from computers and databases elsewhere in the factory. Up to now components of a work cell have been hard-wired together such that a particular wire carried information only between two predetermined machines or controllers. Communications networks are required where information is sent to different receivers over the same wire. The information must therefore be addressed to the receiver, and a return address must be included. The sender must be given appraisal of the status of the message. Was it received correctly? Must it be retransmitted? Information may not be allowed to collide in the network. Several techniques are used to provide collision protection. One method is termed Carrier Sense Multiple Access/Collision Detection (CSMA/CD). Another method is the token-passing scheme used to prevent data collision in the MAP network specification described below. Automated decisions will be based on data communicated over such networks, the decisions will be transmitted over the networks, and updated status will feed back to the decision-making entity over the network. Such a network has great demands placed on it to maintain the integrity of the data during times of massive throughput, to recover from errors and crashes, and to dependably respond in near real time.

Such a network specification, known as the Manufacturing Automation Protocol (MAP), is now under development by General Motors; it is supported by a confederation of over 100 vendors, universities, and government agencies. MAP is designed to connect heterogeneous computer, robotic, and machine tool systems from different manufacturers that were not specifically designed to work together. Although the MAP standard is based on the seven-layer Open Systems Interconnect (OSI) model of the International Standards Organization, fewer than half of the levels are currently defined by accepted standards. Researchers at GM, NBS, and elsewhere are trying to develop the specification for each level in such a way that the standard will be able to evolve and adapt to future technological advances. Networks such as MAP will make IRs smarter by giving robot controllers access to programs, data, and services on other parts of the network and on other networks.

As manufacturing networks improve and standardize, the physical and logical paths for communication to and from IRs will be in place. More than networks will be required, however, for the integrated sharing of information between IRs and other computerized entities. Standards are required for data structures that contain complete descriptions of parts and processes so that the robot can reliably interpret the data that was entered at a CAD terminal, enhanced at a CAE (computer-aided engineering) terminal, and stored in a manufacturing database. This is not a trivial task. One promising approach to the data-structures problem is taken by researchers at Hughes Aircraft Co. and Arizona State University working on GT systems. Tomorrow's robots will have ready access to more data,

programming, and decision-making support through manufacturing communications systems now under development.

### Future Applications

Experts in the field of AI have conflicting predictions on the future effect of AI on robotics and manufacturing (16). Bowden and Winograd, e.g., expect the continued development of domain-specific knowledge systems, but they do not expect AI systems to exhibit general intelligence or common sense. Newell suggests that AI may gain more from its exposure to robotics than robotics gains from AI because robots force AI to accommodate to real-world uncertainties. Michie and Nilsson forecast a new industrial revolution spawned by AI applications in robotics and manufacturing.

The largest projected applications for robots lie in manufacturing. More robots will be used for tasks of lower constraint such as arc welding and assembly (44). The new tasks will require the use of complex sensors such as vision systems to guide the robot and to provide real-time feedback as to the state of the task. Through the use of tactile sensors, IRs will be able to work with a greater variety of materials from fabric to fruit. With access to extensive libraries of programs and descriptions of parts and processes, IRs will be more flexible than ever before. The current trend of combining work cells into a flexible manufacturing system (FMS), and of including FMSs within a larger CIM system, will continue to gain acceptance and momentum. Most IRs will work within the context of such a computer-integrated system. Some vendors will specialize in the custom design and fabrication of complete production subsystems that include many IRs; Penn Field Industries, Inc., is one pioneer in this area.

The applications of intelligent robots will expand beyond the manufacturing arena, but this will be a slow process due to the unconstrained environment problem. Medical robotic devices will be used for surgery that involves actions more precise than the human hand reliably allows and to provide the handicapped with arms, hands, legs, voices, and sight. Robots for the handicapped may be intimately integrated into the human system much as IRs can be integrated into a manufacturing system.

Military robots will be required to operate in environments of low constraint. It is too early to predict the success of current ideas, but basic research is progressing at universities like CMU. Robots that can explore and mine resources in areas that are inaccessible to humans will not be a reality in the near future. Robots will soon be able to operate in a limited way on other planets, but they will not be economically practical as miners or prospectors. Robots to explore space or to mine the sea floor suffer from the same difficulties of operating in an unconstrained environment as do military robots (45).

Practical personal robots will be no more than a curiosity for years because of the unconstrained nature of personal living quarters. Dramatic progress may occur, however, if living quarters are optimally designed for the robot. For example, a robot could cook and clean up if the refrigerator, range, sink, utensils, food packaging, and kitchen layout were designed to be used primarily by the robot. Completely autonomous anthropomorphic robots, such as Isaac Asimov's fictional android, Daneel, may remain the province of science fiction forever.

Robots will increase in intelligence. They will become more adaptable and flexible, with improved decision-making capa-

bilities and almost instant access to required information. The proliferation of robots will continue to gain momentum, particularly in manufacturing applications, where they will take over many repetitive tasks that contain irregularities that heretofore have required the adaptability of humans. Other applications will develop more slowly owing to the difficult problem of programming a robot to deal with the anomalies found in relatively unconstrained environments.

### BIBLIOGRAPHY

1. W. B. Gevarter, "An Overview of Artificial Intelligence and Robotics," Report NBSIR 82-2479, National Bureau of Standards, 1982.
2. R. A. Lewis and A. K. Bejczy, Planning Considerations for a Roaming Robot with Arm, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford University, 1973.
3. T. Lozano-Perez, "Robotics," *Artif. Intell.* **19**(2), 137-143 (1982).
4. R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artif. Intell.* **2**(2-4), 189-208 (1971).
5. R. E. Fikes, P. E. Hart, and N. J. Nilsson, "Learning and executing generalized robot plans," *Artif. Intell.* **3**(1-4), 251-288 (1972).
6. M. P. Groover, M. Weiss, R. N. Nagel, and N. G. Odrey, *Industrial Robotics: Technology, Programming, and Applications*, McGraw-Hill, New York, 1985.
7. B. Shimano et al., Unimation, VAL: A Robot Programming and Control System, Unimation, Inc., Danbury, CT, 1977.
8. Robot Institute of America, *Worldwide Robotics Survey and Directory*, Society of Manufacturing Engineers, Dearborn, MI, 1983.
9. H. L. Martin and D. P. Kuban (eds.), *Teleoperated Robotics in Hostile Environments*, Robotics International of SME, Publications Development Department, Marketing Services Division, Dearborn, MI, Manufacturing update series, 1987.
10. T. Lozano-Perez, "Compliance in robot manipulation," *Artif. Intell.* **25**(1), 5-12 (1985).
11. M. H. Raibert, H. B. Brown, E. H. Chepponis, J. Koechling, K. N. Murphy, S. S. Murthy, and A. J. Stentz, Dynamically Stable Legged Locomotion, Progress Report, October 1982-October 1983, Robotics Institute, Carnegie-Mellon University, 1983.
12. D. J. Todd, *Walking Machines: An Introduction to Legged Robots*, Chapman and Hall, New York, 1985.
13. A. J. Barbera, M. L. Fitzgerald, J. S. Albus, and L. S. Haynes, RCS: The NBS Real-Time Control System, *Robots 8 Conference Proceedings*, Robotics International of SME, Dearborn, MI, pp. 1-33, June 1984.
14. F. E. Acker, I. A. Ince, and B. D. Ottinger, TROIKABOT: A Multiarmed Assembly Robot, *Robots 9 Conference Proceedings*, Robotics International of SME, Dearborn, MI, pp. 1-19, 1985.
15. S. J. Hawker, R. N. Nagel, R. Roberts, and N. G. Odrey, "Multiple robotic manipulators," *BYTE* **11**(1), 203-219 (January 1986).
16. D. G. Bobrow and P. J. Hayes, "Artificial intelligence: Where are we?" *Artif. Intell.* **25**, 375-415 (1985).
17. W. J. Kerth, Jr., Knowledge-Based Expert Welding, *Robots 9 Conference Proceedings*, Robotics International of SME, Dearborn, MI, pp. 98-110, 1985.
18. C. Blume and W. Jakob, *PASRO: Pascal for Robots*, Springer-Verlag, New York, 1985.
19. R. A. Volz and T. N. Mudge, Robots Are (Nothing More Than) Abstract Data Types, *Robotics Research: The Next Five Years and Beyond*, Society of Manufacturing Engineers, Dearborn, MI, MS84-493, pp. 1-16, 1984.

20. R. H. Taylor, The Synthesis of Manipulator Control Programs from Task-Level Specifications, Technical Report AIM-282, Stanford University, 1976.
  21. S. Mojitaba and R. Goodman, AL User's Manual, Memo AIM-323, Stanford Artificial Intelligence Laboratory (SAIL), 1979.
  22. L. I. Lieberman and M. A. Wesley, "AUTOPASS: An automatic programming system for computer controlled mechanical assembly," *IBM J. Res. Dev.* **21**(4), 321-333 (1977).
  23. R. J. Popplestone, A. P. Ambler, and I. M. Bellos, *An Interpreter for a Language for Describing Assemblies*, MIT Press, Cambridge, MA, pp. 537-566, 1982.
  24. D. Lenat, "EURISKO: A program that learns new heuristics and domain concepts," *Artif. Intell.* **21**, 61-98 (1983).
  25. W. A. Perkins, A Computer Vision System That Learns to Inspect Parts, Research Publication GMR-3650, General Motors Research Laboratories, 1981.
  26. R. N. Nagel, Robotics: A State of the Art Review, *Proceedings of the Fourth IEEE Jerusalem Conference on Information Technology*, IEEE Computer Society, Silver Spring, MD, pp. 566-577, 1984.
  27. C. P. Day, L. L. Whitcomb, and D. J. Daniel, Gray Scale Vision System for Robotic Applications, *Robots 9 Conference Proceedings*, Robotics International of SME, Dearborn, MI, pp. 1-12, 1985.
  28. R. A. Brooks, *Model-Based Computer Vision*, UMI Research, Ann Arbor, MI, 1984.
  29. K. E. Pennywitt, "Robotic tactile sensing," *BYTE* **11**(1), 177-200 (January 1986).
  30. D. DeRossi and P. Dario, Multiple Sensory Polymeric Transducers for Object Recognition Through Active Touch Exploration, *Robotics Research: The Next Five Years and Beyond*, Society of Manufacturing Engineers, Dearborn, MI, MS84-492, pp. 1-13, 1984.
  31. J. L. Schneiter and T. B. Sheridan, "An optical tactile sensor for manipulators," *Robot. Comput. Integ. Manufac.* **1**(1), 65-71, 1984.
  32. G. E. Neville, Jr., E. F. Schildwachter, and K. L. Doty, *Alternative Skin Geometries and Materials for Induced Vibration Touch Sensors*, University of Florida, Gainesville, FL, 1985.
  33. J. K. Salisbury and J. J. Craig, "Articulated hands: Force control and kinematic issues," *Proceedings of the 1981 Joint Automatic Control Conference*, Charlottesville, VA (New York, IEEE), vol 1, pp. TA-2c/1-9, 1981.
  34. R. N. Nagel and S. R. Garrigan, Robot Software: Current State of the Art, and Future Challenges, *Artificial Intelligence and Robotics*, NATO Advisory Group for Aerospace Research & Development, Lisbon, September 1985.
  35. H. B. Kuntze and W. Schill, Methods for Collision Avoidance in Computer Controlled Industrial Robots, *Proceedings of the Twelfth International Symposium on Industrial Robots*, Paris, France, June 1982, pp. 519-530.
  36. M. Juliere, L. Marce, and H. Place, A Guidance System for a Mobile Robot, *Proceedings of the Thirteenth International Symposium on Industrial Robots, ROBOTS 7*, 1983.
  37. C. C. Jorgensen, W. Hamel, and C. R. Weisbin, "Autonomous robot navigation," *BYTE* **11**(1), 223-235 (January 1986).
  38. T. Lozano-Perez, "Automatic planning of manipulator transfer movements," *IEEE Trans. Syst. Man, Cybernet.* **SMC-11**, 681-689 (1981).
  39. H. P. Moravec, Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover, Technical Report AIT-340, Stanford, 1980.
  40. S. M. Udupa, Collision Detection and Avoidance in Computer Controlled Manipulators, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, MIT Press, Cambridge, MA, pp. 737-748, 1977.
  41. J. L. Crowley, "Navigation for an intelligent mobile robot," *IEEE J. Robot. Autom.* **RA-1**(1), 31-41 (March 1985).
  42. S. S. Iyengar, C. C. Jorgensen, S. V. N. Rao, and C. R. Weisbin, Robot Navigation Algorithms Using Learned Spatial Graphs, Technical Report ONRL-TM-9782, ONRL, 1985.
  43. R. Vernadat, Communication: A Key Requirement in Computer Integrated Manufacturing, *Proceedings of the Second Annual Phoenix Conference on Computers and Communications*, Phoenix, AZ (New York, IEEE), pp. 193-198, 1983.
  44. V. J. Pavone and E. Phillips, Arc Welding Robots: An Overview, *Robots 9 Conference Proceedings*, Robotics International of SME, Dearborn, MI, pp. 111-133, 1985.
  45. R. E. Korf, Space Robotics, Technical Report, Carnegie-Mellon University Robotics Institute, Pittsburgh, PA, 1981.
- General References**
- J. Albus, *People's Capitalism: The Economics of the Robot Revolution*, New World, Kensington, MD, 1976.
- I. Asimov and K. A. Frenkel, *Robots, Machines in Man's Image*, Harmony, New York, 1985.
- G. Beni and S. Hackwood (ed.), *Recent Advances in Robotics*, Wiley, New York, 1985.
- M. Brady and R. Paul (ed.), *Robotics Research: The First International Symposium*, MIT Press, Cambridge, MA, MIT Press series in artificial intelligence, 1984.
- H. Hanafusa and H. Inoue (eds.), *Robotics Research: The Second International Symposium*, MIT Press, Cambridge, MA, MIT Press series in artificial intelligence, 1985.
- Proceedings of the 1983 International Conference on Advanced Robotics*, Robotics Society of Japan, Society of Biomechanisms, and the Japan Industrial Robot Association, Scholium International, Inc. (Great Neck, NY), Tokyo, 1983.
- Proceedings of the International Conference on Robotics*, IEEE Computer Society, Institute of Electrical and Electronics Engineers, New York, 1984.
- IEEE Conference on Robotics and Automation*, IEEE Computer Society, IEEE Computer Society Press, Silver Spring, MD, 1985.
- Y. Koren, *Robotics For Engineers*, McGraw-Hill, New York, 1985.
- M. Minsky (ed.), *Robotics*, Omni, Garden City, NY, 1985.
- Robot Institute of America, *Robot Institute of America Worldwide Robotics Survey and Directory*, Society of Manufacturing Engineers, Dearborn, MI, 1982.
- B. Rooks (ed.), *Developments in Robotics 1983*, IFS (Publications) Ltd., UK, 1983.
- Robots 9 Conference Proceedings*, Society of Manufacturing Engineers, Dearborn, MI, 1985.
- Robots 8 Conference Proceedings*, Society of Manufacturing Engineers, Dearborn, MI, 1984.
- Conference Proceedings of the Twelfth International Symposium on Industrial Robots and Robots 6*, Society of Manufacturing Engineers, Dearborn, MI, 1983.
- Proceedings of the Second Canadian CAD/CAM and Robotics Conference*, The Canadian Institute of Metalworking and the Society of Manufacturing Engineers, SME (Dearborn, MI), Toronto, Canada, 1983.
- Conference Proceedings of the Thirteenth International Symposium on Industrial Robots and Robots 7*, Society of Manufacturing Engineers, Dearborn, MI, 1982.
- Taxonomy and Control**
- M. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M. T. Mason, *Robot Motion: Planning and Control*, MIT Press, Cambridge, MA, 1982.
- M. T. Mason, *Compliant Motion*, MIT Press, Cambridge, MA, pp. 305-322, 1983.

- R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press, Cambridge, MA, 1981.
- T. B. Sheridan, Supervisory Control of Remote Manipulators, Vehicles and Dynamic Processes: Experiments in Command and Display Aiding, Research Report, MIT Man-Machine System Laboratory, Cambridge, MA, 1983.
- J. A. Simpson, R. J. Hocken, and J. A. Albus, "The Automated Manufacturing Research Facility of the National Bureau of Standards," *J. Manufact. Sys.* 1(1), 17-32 (1982).
- D. E. Whitney and others, Design and Control of Adaptable-Programmable Assembly Systems, Report R-1406, Charles Stark Draper Laboratory, Inc., Prepared for NSF Grant No. DAR77-23712, 1981.

#### Sensors

- R. Bracho, J. F. Schlag, and A. C. Sanderson, *POPEYE: A Gray-Level Vision System for Robotics Applications*, The Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, 1983.
- Proceedings of Robotics and Robot Sensing Systems*, SPIE, International Society for Optical Engineering, San Diego, CA, 1983.
- P. Dario and D. De Rossi, "Tactile sensors and the gripping challenge," *IEEE Spectr.* 22(8), 46-52 (August 1985).
- A. Gomersall, *Machine Intelligence: An International Bibliography with Abstracts of Sensors in Automated Manufacturing*, Springer-Verlag, New York, 1984.
- L. D. Harmon, "Automated tactile sensing," *Int. J. Robot. Res.* 1(2), 3-32 (Summer 1982).
- T. Kanade, "Visual sensing and interpretation: The image understanding point of view," *Comput. Mech. Eng.*, 1(4) 59-69 (April 1983).
- R. K. Miller, *Machine Vision for Robotics and Automated Inspection*, Technical Insights, Fort Lee, NJ, 1983.
- H. Mori, Project "Harunobu": Toward a Locomotive Vision Robot, *Proceedings of the Fourth International Conference on Robot Vision and Sensory Controls*, London, U.K., 1984, pp. 477-485.
- R. Nagel, C. Vandenburg, J. Albus, and E. Lowenfeld, "Experiments in part acquisition using robot vision," *Robot. Tod.*, 30-40 (Winter 1980-81).
- D. Nitzan, Assessment of Robotic Sensors, *Proceedings of the First International Conference on Robot Vision Sensory Controls*, Stratford, U.K., April 1981, pp. 1-8.
- Proceedings of the Fourth International Conference on Robot Vision and Sensory Controls*, London, U.K., 1984.
- A. Pugh (ed.), *Robot Vision*, IFS (Publications), U.K., International Trends in Manufacturing Technology, 1983.

#### Programming

- I. Aleksander (ed.), *Computing Techniques for Robots*, Chapman & Hall, New York, 1985.
- D. D. Grossman, "Robotic software," *Mechan. Eng.* 104(8), 46-47 (August 1982).
- K. G. Kempf, Robot Command Languages and Artificial Intelligence, *Robots 6 Conference Proceedings*, Robotics International of SME, Dearborn, MI, pp. 369-391, 1982.
- S. Mojtaba and R. Goodman, AL User's Manual, Memo AIM-323, Stanford Artificial Intelligence Laboratory (SAIL), 1979.
- B. E. Shimano, VAL: A Versatile Third Robot Programming and Control System, *Proceedings of IEEE Third International Symposium on Industrial Robots*, Chicago, 1979, pp. 878-883.

#### Communications & CIM

- D. E. Atkins and R. A. Volz, Coordinated Research in Robotics and Integrated Manufacturing, Annual Report, August 1, 1982-July

31, 1983, Center for Robotics and Integrated Manufacturing, University of Michigan, 1983.

*Robotics and Factories of the Future*, International Conference on Robotics and Factories of the Future, Springer-Verlag (New York), Charlotte, North Carolina, 1984.

M. P. Groover, *Automation, Production Systems, and Computer Aided Manufacturing*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

L. L. Toepperwein and M. T. Blackman, ICAM Robotics Application Guide, Report AFWAL-TR-80-4042, Vol. 2, ICAM Program, USAF, 1980.

#### Artificial Intelligence

- J. S. Albus, *Brains, Behavior, and Robotics*, BYTE, Peterborough, NH, 1981.
- Robotics and Artificial Intelligence*, NATO ASI Series (series F), Springer-Verlag (New York, 1984), Castelvechio Pascoli, Italy, 1983.
- M. L. Brodie, J. Mylopoulos, and J. W. Schmidt (eds.), *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*, Springer-Verlag, New York, 1984.
- The Second Conference on Artificial Intelligence Applications: The Engineering of Knowledge-Based Systems*, IEEE Computer Society, Institute of Electrical and Electronics Engineers, Miami Beach, FL, 1985.
- R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach*, Tioga, Palo Alto, CA, 1983.
- Artificial Intelligence and Information-Control Systems of Robots (proceedings of Third International Conference)*, International Conference on Artificial Intelligence and Information-Control Systems of Robots, North-Holland (New York), Smolenice, Czechoslovakia, 1984.
- P. H. Winston, *Artificial Intelligence*, Addison-Wesley, Reading, MA, 1984.

#### Applications

- Towards the Factory of the Future*, Vol. PED-1, American Society of Mechanical Engineers, New York, 1980.
- R. U. Ayres and S. V. Miller, *Robotics Applications and Social Implications*, Ballinger, Cambridge, MA, 1983.
- J. F. Engelberger, *Robotics In Practice*, AMACOM, New York, 1980.
- J. Jablonowski, "Robots that weld," *Am. Machin.* 127(4), 113-128 (April, 1983), Special Report 753.
- R. E. Korf, Space Robotics, Technical Report, Carnegie-Mellon University Robotics Institute, Pittsburgh, PA, 1981.
- J. E. Long and T. J. Healy (eds.), *Advanced Automation For Space Missions*, Technical Summary, NASA/University of California at Santa Clara, 1980.
- R. H. Miller, M. L. Minsky, and D. B. S. Smith, Space Applications of Automation, Robotics, and Machine Intelligence Systems (ARAMIS), Technical Report NASA CR-162079-162082, National Aeronautics and Space Administration, 1982.
- H. P. Moravec, The Endless Frontier and the Thinking Machine, Stanford Artificial Intelligence Lab (SAIL), 1978.
- Robot Technology and Applications: Proceedings of the First Robotics Europe Conference*, Robotics Europe, Springer-Verlag (New York), Brussels, Belgium, 1984.
- H. J. Warnecke and J. Walter, Automatic Assembly: State-of-the-Art, *Proceedings of the Third International Conference of Assembly Automation*, 1982, pp. 1-14.

R. N. NAGEL AND S. GARRIGAN  
Lehigh University



## ROBOTS, MOBILE

A mobile robot is a free-roving collection of functions primarily focused on reaching a designated location in space. In the simplest of environments the robot requires at least one or more modes of mobility and reliable knowledge of its position relative to the goal. The robot requires more capability as the environment becomes more complex. Local constraints to free motion toward the goal necessitate a route-finding capability. Route finding often requires sensors even though obstacle maps are available because of the uncertainties in the precision and completeness of the maps. Transit in very complex environments requires a route-planning capability as well. Once the robot reaches its destination docking or manipulation activity may be necessary. All of these functions must be supported by some combination of computing resources.

Although several mobile-robot implementations are discussed in this entry, these examples are not all-inclusive. However, they do fairly represent activity in the field. Other surveys are available that provide additional implementation examples and better historical perspective (1) (see also Autonomous vehicles).

### Mobility

Robot mobility can be broadly divided into motion in a volume medium and motion constrained to surfaces.

Volume task complexity varies from moving freely through empty space toward a goal to moving in close quarters amidst complex structures. The different forms of volume mobility correspond to those available in existing manned or remotely controlled vehicles. Although applications for volume robots include airborne, space, and undersea domains, the most advanced of these are undersea mobile robots. Several examples of these can be found in reviews of robot submersible technology (2,3). Offshore structure inspection and repair robots incorporate the most sophisticated technology. Both teleoperated (4) (see Teleoperators) and semiautonomous (5) submersibles have been demonstrated in this application. Vehicles working around structures require the greatest number of degrees of freedom (DOFs) (6) and implementations have three translational and one (4) or two (5) rotational DOFs. These robots are controlled by thrusters that are usually arranged to decouple vertical and horizontal motions, thus simplifying vehicle control. In addition, vehicle dynamics-based prediction models can make vehicle control more robust by minimizing the effects of measurement noise (4). A hierarchical control strategy isolates low-level controls requiring rapid independent response from higher level slower processes (4,5). Several problems confront surface mobility: translational motion, rotational motion, uneven surfaces, and incomplete surfaces. Surface mobility requires at least one degree of translational motion and some form of propulsion. Electric motors propel many mobile robots from both offboard (4,7-10) and onboard (5,11-27) electrical energy supplies. Some mobile robots use onboard fossil-fueled engines for propulsion and electrical power (23,28,29). Numerous human-controlled surface vehicles for a variety of applications offer plentiful retrofit opportunities. Surface robots generally require some form of suspension to assure the continual contact with the constraining surface necessary for motion and directional control. Mobile robots have used passive, semiactive, and active suspen-

sion systems. Robots moving at low speed on purely planar surfaces require only the crudest form of passive suspension, which often is provided by the inherent flexibility of the chassis. As the constraining surface deviates from a plane, the suspension system complexity must increase until finally evolving into a fully articulated or legged locomotion system.

**Passive Suspension.** Passively suspended vehicles must use some form of wheel or track for mobility. Three-wheeled (14,20,21) four-wheeled (7,11,13,16,17,28), six-wheeled (23), and tracked (29) robots have been implemented. A mobile robot operating on a well-prepared surface with only one translational DOF requires the simplest suspension system. Unfortunately, these robots are inflexible, costly to modify and, thus, find limited application. Some mobile robots combine one translational DOF with one rotational DOF that is often coupled to translational motion (7,23,28,29). Only a few mobility designs offer more than one independent translational DOF (11,14,21). However, uncoupled translational and rotational DOFs permit rotating the vehicle during purely translational motion to gain improved mechanical advantage (21). Mobile robots commonly use differential steering for rotational motion (11,16,17,19,23,24,27,29,30), although other designs include tricycle steering (20,23), two-axle or single-Ackerman steering (29), and four-axle or double-Ackerman steering (7). Increasing the number of steered axles increases the number of independent DOFs but also significantly increases control law complexity (7,14). Several sophisticated steering modes have been developed, but most suffer from considerable control difficulties (14).

**Semiactive Suspension.** Some situations require greater vehicle agility than is provided by passive suspension. Controlling some of the suspension DOFs can provide this agility. Semiactive suspension has been designed for robots for complex indoor environments (8) as well as planetary-surface exploration (9,13). A mobile robot has been developed with two DOF articulated struts and wheels equipped with independently controlled peripheral rollers that can climb stairs and turn in narrow areas for nuclear-plant inspection and repair (8). Planetary explorers require complex suspension in order to reliably negotiate unknown irregular surfaces. One such vehicle had four independently controlled suspension arms and could cross highly irregular terrain and directly overcome obstacles less than 12 in. (0.3 m) high (13). Another planetary-explorer concept incorporated four independently driven loop wheels connected to the vehicle through two DOFs electrically adjustable struts (9).

**Legged Locomotion.** Legged vehicles are the ultimate realizations of active suspension and enable motion over discontinuous, nonhomogeneous, and highly irregular surfaces. A key difficulty in legged locomotion is achieving smooth and stable vehicle motion in the desired direction. Both statically stable and dynamically stable legged vehicles are possible.

A statically stable vehicle maintains its center of gravity within the convex hull of the ground contact points of at least three legs at all times. Efficient control of leg motion remains a major challenge in statically stable legged locomotion (9). Several leg designs have been developed that efficiently decouple the leg motion into vertical and horizontal components (9,12,31). These articulators commonly have three DOFs

(9,12). Linear programming has been used to eliminate parasitic internal forces in leg motion (9). Statically stable vehicles with four (31) and six (9,12) legs with both longitudinally (9,31) and vertically (12) symmetric body geometries have been demonstrated. The optimal solutions for stable leg cycling or gaits are known for straight-line motion over smooth continuous surfaces. Movement across nonhomogeneous or irregular surfaces requires aperiodic gaits. A heuristic algorithm has been developed for aperiodic leg movement that maximizes the number of legs in the air and the amount of time a leg stays on the ground once placed. This strategy sacrifices vehicle stability for improved adaptability and deadlock protection (9). Vehicle stability during movement across irregular surfaces can be improved by adding inertial, leg force, and terrain preview sensing (9,31). Some degree of leg compliance is required to accommodate imperfect modeling of terrain contours and nonideal functioning of joint servomechanisms. This compliance can be passive, active, or both (9).

At higher vehicle speeds dynamically stable locomotion is more efficient than statically stable locomotion (32). Dynamic stability takes advantage of leg elasticity (10). Leg motion is usually divided into flight and support phases (10,32). In dynamically stable locomotion a major portion of vehicle control must occur over a small fraction of the gait cycle (32). Decoupling control of the vertical and horizontal DOFs can simplify solution of the vehicle motion equations (10,32). Dynamically stable legged vehicles of one, two, and four legs have been discussed (10,32). For a one-legged hopping vehicle reasonably simple algorithms can effectively control its forward motion and body angle. These algorithms control the hop energy to control the hopping height and control balance and forward travel by controlling foot placement and hip travel (10). Multi-legged dynamically stable vehicles also have different gaits. Control of three gaits (i.e., amble, trot, and gallop) has been developed for a four-legged vehicle traveling on a flat horizontal surface. Control of this vehicle is simplified by assuming that different body angles are zero for different gaits and by decoupling the control of nonzero components (32).

### Position Finding

Several surveys of position-finding techniques applicable to mobile robots are available (20,33–35). These techniques compute robot position from information about either robot motion or the relative positions of external cues for which the absolute position is known.

Robot motion can be used to find relative position with either dead reckoning or inertial navigation. Dead reckoning is the most common technique because of its simplicity and inexpensive short-term accuracy (5,8,14,22,25,27–30,36–40). Robots can sense translational motion with wheel revolution encoders (14,22,25,27,29,30,36–40), wheel velocity sensors (28), onboard odometers (7,23,28,30), speed prediction using previous absolute position information (5), and Doppler true ground-speed sensors (29). Robots can measure rotational motion by computing differential wheel rotation from wheel encoders (14,22,25,27,29,30,36–40), by computing differential wheel velocity from wheel-velocity sensors (28), and from a compass (5,7,23,28,29). Dead reckoning has the following problems: wheel slippage generates unpredictable errors and errors accumulate unless the position fix is updated. Doppler speed sensing circumvents wheel slippage errors but is expensive (5). Nevertheless, dead reckoning can be inexpensive, has reasonable short-term accuracy, and the information is in-

stantly available in any situation (20,33). Inertial navigation systems also have good short-term accuracy, provide instantaneous availability, and are easy to implement, but they are also very expensive, usually bulky, drift with time, and have accumulating errors (5, 33). Very few mobile robots have actually used inertial navigation.

**Beacon Systems.** Beacon systems all triangulate an active transmission for position finding. Beacon systems have been suggested for automated vehicle location (34). A carefully designed beacon system can often provide both vehicle position and orientation (5). The triangulation process uses either time or phase information (34). Both active and passive vehicle configurations have been discussed (34), but most mobile robots employ passive vehicle techniques (4,5,29,38). Common beacon types include acoustic transponders for undersea (4,5), infrared beacons for indoors (38,40), fixed radio transmitter sites (34), and earth-orbiting satellites for outdoors (29). Although beacon systems offer very good long-term and, in some cases, short-term accuracy, position measurements may be available only at certain locations due to shadowing (5,20), and position updates may come slowly (5). In addition, some rf and acoustic systems suffer from multipath problems (5,34). Systems using active vehicles have the added disadvantage of needing communication with a base station (34).

**Landmark Systems.** Several techniques have been suggested to determine absolute position by observing external passive landmarks (35). Landmark position finding can use contrived signals (36,41,42), local structural features (18,22,38,40,43), and natural landmarks (35,44,45) as reference points. The simplest technique uses a signal of known shape and size and a vehicle-mounted camera (36,41,42). This technique is highly dependent on the diagonal width of the signal; it does not work for all orientations, and it does not work when the distance between the vehicle and the signal is either too large or too small (42). Techniques have been developed to match maps of local structural features to perceptions from vision (18,43), laser range finder (38,40), and ultrasonic range finder (22,27,38). Robot-motion information derived from optical parameters (43,45) and dead reckoning (18,38,40) can predict expected position to minimize search for matching. The matching search can be further minimized using visibility maps (40). Moving and unexpected objects can be handled by simulating the motion of expected objects with predictable movements (e.g., doors) (18). The visual-mapping problem in indoor environments can be simplified by matching only vertical lines (18,24). In general, visual-matching techniques include sparse 2-D matching, continuous 2-D matching, and 3-D matching (45). In addition, fast, stable, and robust techniques have been developed for visual map matching using relaxation instead of search (43,45). Visual techniques tend to match object vertices (18,24,38,40,43), and acoustic techniques match object surfaces (22,38). The problems with imaging techniques include complex time-consuming processing and critical dependence on the ambient lighting conditions. Ultrasonic ranging systems, on the other hand, provide fuzzy images, are temperature-dependent, and suffer from multiple reflections (33).

In order to avoid the shortcomings of any one source of position information, many mobile robots use various combinations of the above techniques. Some use dead reckoning updated with a beacon system (5,14,29,34,36,38,40), and others use dead reckoning with sonar (22,38) or laser (38,40) map matching to refine position estimates. Once an improved posi-

tion estimate is available from local map matching, the rest of the map can be updated with improved past position estimates by propagating the vehicle odometry error backward through the robot path and recomputing the past position errors (38). Some mobile robots use at least three sources of position information including dead reckoning, beacons, and map matching (29,38,40). In these systems trajectories are executed using dead-reckoning measurements updated from either map-matching estimates when an obstacle is passed or absolute position measurements from beacons when they are available (40).

### Route Finding

Several techniques exist that permit a mobile robot to find its way to its goal using sensors when a direct path is not available. The simplest techniques take advantage of fixed prescribed routes. More complex techniques have demonstrated road following, and the most challenging route-finding problem involves obstacle avoidance.

**Preprogrammed Paths.** Preprogrammed path guidance provides the simplest and most robust technique for route finding. In the industrial domain robot paths are marked with either buried wires or painted lines (33). Robot-control signals are derived directly from wire (coils) or painted-line (photodetectors) position sensors. Buried-wire systems permit different routes to be defined with different signal frequencies. Sophisticated buried-wire systems can accommodate corners and communicate routing data to multiple robots through the cable. Buried-wire systems are popular in industry and fairly reliable (15) but are inflexible and expensive (33). Painted-line systems overcome these problems, but painted-line networks must be simple, junctions are not easily handled, the lines can be obscured by objects, and the lines need regular repainting. These problems make painted-line guidance impractical for medium to heavy industry, but they remain ideal for light engineering and office environments (33). Visual recognition of painted lines using a solid-state camera and image processing has also been discussed (46). Another simple path-following technique uses precise knowledge of position to follow predetermined routes in space (18,20,24,40,43).

**Structure Following.** Beyond simple path following is structure following. Several robots have demonstrated wall following using fixed (19,38) and steered (24,26) ultrasonic range finders. Some of these vehicles can even follow walls in the presence of obstacles (19,26). A farming robot uses ultrasonic range finders to follow plowed furrows (17). Ultrasonic ranging sensor performance is good for wall following in the absence of irregular reflecting surfaces (e.g., doorways, corners) (24). In another approach for an automated lawnmower, a linear array camera is used to detect and follow the grass cut line. This capability enables the vehicle to follow one of several available preprogrammed cutting strategies (e.g., strip, perimeter, sector) (36). Some mobile robots can recognize and follow moving targets (27,29).

**Road Following.** Several automated road-finding techniques using digitized images have been developed (23,28,43,44,47), and some of them have been demonstrated on mobile robots (23,28,43). These techniques have used imagery from both single (43,47) and stereo (23,28) cameras for both road following and obstacle avoidance. Single-camera imple-

mentations must assume some scene structure (e.g., flat roads) (47). Almost any edge-detection technique works for simple situations (i.e., high contrast, straight lines, low scene complexity). However, existing techniques have proved inadequate for complex outdoor scenes (23). Several different line-extraction techniques have been used including least-squares fitting (23,29), line tracing (23), and modified Hough transform (qv) (23,43,47). Road-edge search can be constrained using knowledge of vehicle position and previous road location (23,43). The vehicle is usually controlled by steering through computed target points that follow the road as they avoid obstacles (23,28). Mobile robots have been demonstrated following roads at speeds of 0.02 m/s along an obstacle-free road (23), 2.8 m/s along a road with obstacles (28), and 0.4 m/s through the hallways of a building (43).

**Obstacle Avoidance.** In the simplest situation the robot has a perfect map locating all obstacles in the environment. Several mobile robots use map information (18,24,25,43). Unfortunately, complete maps are not always available, and then robot sensors must be used to gather additional route information (7,18,25). Efforts using maps for obstacle avoidance often employ route-planning techniques to determine the robot's exact route and then navigate that route using precise position-finding techniques. The obstacle-avoidance sensors include single- (18,19,25,29) and multiple-camera systems (7,21,23,27,28), infrared proximity sensors (21,27) and ultrasonic (16,19,21,23,26,27,29,36,39) and laser range finders (7,13,19,29,40). All obstacle-avoidance techniques require determining the obstacle range from the vehicle. Several range-finding techniques are available (48). No visual technique is better than another (48), and these are all limited by complex time-consuming scene analysis and registration procedures (33), critical dependence on ambient lighting conditions (28,33), and ambiguities of interpretation arising from the occasional lack of correspondence between object boundaries and inhomogeneities of intensity, texture, and color (48). Capturing the third dimension with nonimaging techniques usefully avoids these problems (48).

Several mobile robots take advantage of ultrasonic-ranging sensors in fixed (19,21,23), steered (26,36,39), and combination (16,27,29) configurations. Ultrasonic-sensor-bearing resolution has been increased using overlapping sensors (23,27) and by steering the sensors (16,26,27,29,36,39). Problems with ultrasonic ranging include fuzzy images, temperature dependence, and multiple reflections (33). Laser range finders with an intense enough energy source avoid all drawbacks of visual and acoustic-ranging systems, but this solution is often expensive and hazardous to humans (48). Both time-of-flight (7,29) and triangulation (13,19,40) laser sensors are applicable to obstacle detection although time-of-flight sensors are generally better for long ranges and triangulation better for short ranges (13). Most mobile robots use many sensors simultaneously for obstacle identification and avoidance including combinations of contact sensors (19,27,36), infrared proximity sensors (21,27), acoustic-ranging sensors (19,21,23,27,29,36), 2-D vision (19,29), stereo vision (7,21,23,27) and laser-ranging sensors (7,19,29).

### Route Planning

With a map or sufficient sensor information, a mobile robot can avoid trial and error or maze-walking alternatives by planning its route to some future position. The planning pro-

cess becomes very complicated as the environment becomes more complex, thus requiring planner structuring. This structuring also aids in decomposing the interpretation and planning tasks. The simplest planner consists of a terrain modeler for terrain data interpretation and a path selector (13). The robot's performance can be improved by adding a predictive component (49). Predictive information hastens such complex sensor data processing as stereo matching (43), enables slower modeling and planning mechanisms to keep pace with faster control and sensor-data-acquisition processes (49), and permits control directly from sensor-error signals (43). More complex planning systems are partitioned hierarchically into several levels to isolate processes with different spatial and temporal requirements from one another. Planning has been partitioned by sensor ranges [e.g., long, intermediate, and short (44)], function [e.g., perception, cartographer, and control (50)], and environment organization [e.g., building, room, and position levels (39)], among others (see Planning).

**Sensor Data.** Like planners, models built from sensor data are often organized hierarchically (44). Supplementary model organizations have been suggested [e.g., geometric, topological, and semantic (40)]. Regardless of the sensor-data organization, a representation of free space usually results. The representation's form depends on the sensors used to collect the information (e.g., acoustic-range data are represented as connected sequences of line segments (22,38,39) and vision (43,50) and laser-range data (40) are represented as object vertices in two (40,50) and three (43) dimensions. The most recent and visible sensor contacts can be merged into a single sensor map using various admittance criteria (22).

**Maps.** Maps represent sensor data and a priori knowledge with new sensor data added as it becomes available (51). Many robots divide maps into convex polygons that represent passable, impassable (7,22,38,39,51), and unknown (7,51) regions. Maximizing region sizes minimizes route-planning computation (22). These regions can be labeled with such attributes as region type (18), traversability (50), and dynamic properties (18). Route planning can be simplified by shrinking the robot to a dimensionless point and expanding the sizes of obstacles on the map (25,39,50). Uncertainty in maps has been represented using states (22) and probabilities (38). If probabilities are used, sensor data may be combined using Bayes's rule (see Bayesian decision methods) or some other combination formalism (38).

**Path Search.** Maps are generally transformed into path trees (39) or graphs (22,38–40,50,51) that are then searched for optimal paths. The links of the path graph may represent distance or some more complex costs [e.g., risk (51), traversability (50), route complexity (28)]. This search (qv) process often involves either tree search (39) or graph search (22,28,39,50,51). In tree search expanding and pruning techniques reduce the complexity of the search and can be combined with heuristics to minimize needless backtracking (39). Graph-search techniques such as a version of Dijkstra's algorithm (22), modified A\* search (40) (see A\* algorithm), breadth-first search (39), the Wandering standpoint algorithm (50), and a version of Bellman's dynamic-programming algorithm (51) have been used. If a complete path to the goal cannot be identified, the robot must use motion to acquire sufficient information to plan the route to the goal. This motion may take the robot to the boundaries of unknown areas (38,51) or to better observation points (51). Spline interpolation can predict

the character of unknown areas, but this technique works best for prediction of terrain geometry and fails for hazardous areas such as marshes, embankments, and rivers (51). Unknown natural terrain requires such special capabilities as temporal planning, planning in dynamic worlds, planning using non-geometric parameters, and 3-D modeling for path planning (52). Production systems permit complex heuristics to be easily incorporated into the planning process. These heuristics can be employed as domain-specific and domain-independent plan critics that attempt to reduce plan errors before execution (39,53) and in plan-execution monitoring and repair when unexpected situations are encountered (39). Compiling rules into a parallel match tree greatly improves production-system performance (53).

### Computing Architectures

Computing architectures for mobile robots evolved with the development of the microprocessor. Early robots (13) and robots demonstrating complex locomotion schemes (9,39) used completely remote computing facilities accessed through a cable (9,39) or a rf telemetry link (13). When microprocessors became available, they were employed as onboard computing for mobile robots. Early microprocessors had to be supplemented with remote computing and several mobile robots coupled a single onboard microprocessor to a remote mainframe (4,7,16,18). The onboard processor handles such tasks as sensor-data collection and vehicle control. The control information transmitted to the vehicle can be reduced by grouping motor functions into high-level commands, using reflex circuits for quick local control, and using intelligent peripheral controllers (i.e., devices built to execute a specialized instruction set for a particular control activity (16).

More complex onboard multiprocessor computing for mobile robots has taken several forms. The simplest is a master-slave arrangement between onboard processors (16,17). A special master-slave organization consists of an analog front end for master preprocessing [e.g., laser spot finding (40), stereo image comparison (28), and vertical edge finding (18,24)] with digital post-processing. Beyond the master-slave organization is a simple star architecture with a central computer coordinating functionally partitioned processors (5,12,19,23,26,27,36,37,40,54,55). More complex systems can be organized as a hierarchy of stars or a tree (27,36,37,54). Trees with from 3 to 16 processors have been implemented in from two to four hierarchical levels. Usually the higher levels integrate the low-level sensor data and provide commands to the low-level controllers (5,27,36). The fastest processes are executed on the lowest levels of the tree (5,19,23,27,36,55). Many systems continue to use offboard mainframe computing resources for such complex tasks as planning (19,23) and vision interpretation (23,37), and others are completely self-contained (5,26,27,29,36). Some systems communicate through message passing over a common bus (27,36,54) and others use shared memory (7,43). Tasks on low-level processors can be organized as loosely coupled processes communicating through message passing (26) or as cooperating experts communicating through a blackboard (37) (see Blackboard systems). Processors organized as a set of functionally partitioned and loosely coupled processes communicating through either a parallel (24) or a serial bus (29,56) provide an alternative to star and tree architectures. In this arrangement the processors are made as independent as possible and communicate a minimum amount (26,49). A cellular-array (SIMD) computing sys-

tem has been applied to mobile robot path planning and has demonstrated planning with only a linear dependence on number of obstacles and greatly increased path-generation speed (25). Several have employed multitasking operating systems to coordinate onboard computing resources (7,24,26,29). Mobile robot computers have been programmed in such languages as FORTH (36), ASSEMBLY (27), VAL (14), and PL/M (29).

### Destination Activities

**Docking.** Once the robot arrives at its destination, it often needs to dock with another object by making mechanical (14,15) and often electrical (14,27) and communications (14) connections. In some cases it is necessary to use beacons attached to the docking area for the rendezvous (14,27). The robot can more easily extract the beacon signal from background illumination if it modulates the beacon through a communications link (27). Firm mechanical connection to the dock provides both precise position location and stability (14,15). The docking apparatus can be placed at either waist (14) or floor (15,27) levels.

**Manipulation.** Manipulators (qv) may be needed upon arriving at the destination. Manipulation capability ranging from a simple motor-driven claw (16) through 3-DOF (12,17,21), 5-DOF (15), 6-DOF (11,14,15), and 7-DOF (7) manipulators have been suggested for mobile robots. Manipulator subsystem complexity varies from a legged vehicle using one specially equipped leg as a lifting hook (12) to a robot with two arms mounted on a 2-DOF waist (11). Some mobile robots use commercially available industrial robots (14,15). Manipulators have been equipped with and coordinated through several sensor types. Teleoperated robots with manipulators have been operated through cameras (11,17), and automated vision has been demonstrated coordinating a manipulator on an autonomous mobile robot (7). Mobile-robot manipulators have also been equipped with tactile (7,17) and proximity sensors (7). Mobile-robot manipulators have been used for harvesting (17), material handling (7,15), and digging (7). The manipulator can be used for offboard work over extended task envelopes or for loading and unloading (15). In some cases the mobility of the vehicle can reduce the necessary DOFs of the arm (e.g., shoulder joint) (21). It has been suggested that all active components for loading, unloading, and docking be located on the vehicle and only passive components on the work stations (15).

### Applications

Mobile robots have been suggested for industrial, nuclear, undersea, military, and space applications.

**Industrial.** The amortization of the high cost of industrial automation depends on full utilization. Full utilization is not possible without the ability to transport material and workpieces from workcell to workcell (15). This is the key role that mobile robots play in the factory domain. Flexible mobile robots help with load balancing at fixed workcell sites and enable better use of space since workcell sites need not be laid out to minimize the distance between related operations (15). Wire-guided vehicles were used first in storage and commissioning operations as adaptable transport devices for workpiece and material transport (15). Contemporary factory uses for mobile robots include machine loading and servicing, clean

room-material handling, routine cleaning and maintenance, greenhouse-plant care (14), workpiece supply of lathes and manufacturing cells, tool handling (although reaction forces may present problems), and welding or coating large structures (15). Mobile robots can simply carry material from one workcell to another (15) or they can service multiple work stations (14). A key requirement for wider industrial application is the ability to change the robot's route easily (15). Experimental mobile robots have been developed for harvesting white asparagus (17), mowing lawns (36), servicing multiple work stations (14), monitoring building interiors (18), mining (14), and material and tool transport (15).

**Nuclear.** Mobile robots are becoming necessary for various remote operations such as maintenance (39) and repair (11) tasks in nuclear facilities (39). Several experimental teleoperated mobile robots have been developed for nuclear applications (11). In addition, legged robots have been suggested for nuclear-plant surveillance, maintenance, decontamination, accident recovery, refurbishing, decommissioning, and waste disposal (12).

**Undersea.** The remotely controlled submersible is now an established offshore engineering tool (4). Several different vehicles and sensor systems have been developed for undersea applications (3). Three different generic configurations are being developed in robot submersibles that correspond to three different work domains (i.e., long-distance exploration, sea-floor activities, and underwater-structure inspection and repair) (6). Remotely operated vehicles have been used for such applications as pipeline inspection, underwater-structure inspection, underwater-structure cleaning, sediment classification, cable detection and tracking, undersea-well-head inspection and maintenance, undersea surveying, and underice surveying (3,5).

**Military.** Several reasons have been proposed for employing mobile robots in combat: freedom from fear, mobility, low cost, high mission payload, stamina, modularity, low-life-cycle cost, and expendability. It is not necessary to have humanlike intelligence (57). Work in the areas of multisensor integration, new power sources and propulsion, communications and remote management, and robot brain (both software and computers) are required before capable battlefield robots can be realized (57). Potential military applications for mobile robots include reconnaissance, explosive ordinance disposal, battlefield surveillance (1,58), sentry, patrol, hazardous travel, decoy, mine laying, mine detection, search and rescue, weapons deployment, intelligence collection, and electronic warfare.

**Space.** Mobile robots have been suggested for planetary exploration both from space and on the planet's surface (13,59). Interstellar navigation could be possible with an autonomous probe. A potential near-term space mission for a mobile robot would be a Titan explorer (59).

### BIBLIOGRAPHY

1. B. Schacter, G. Tisdale, and G. Jones, Robot Vehicles: A Survey and Proposed Test-bed Facility, paper presented at DARPA Image Understanding Workshop, Washington, DC, July 1983.
2. A. Westneat, D. Blidberg, and R. Corell, "Advances in unmanned untethered underwater vehicles," *Unman. Syst.* 1(3), 8-13 (Winter 1983).



3. R. Wernli and J. Jaeger, ROV Technology Update from an International Perspective, *IEEE Oceans '84*, 84CH2066, Washington, DC, September 1984, 10-12, pp. 639-645.
4. G. Russell, The Automatic Guidance and Control of an Unmanned Submersible, in A. Pugh (ed.), *Robotic Technology*, Peregrinus, Exeter, U.K., pp. 52-62, 1983.
5. D. Blidberg, An Underwater Automation Employing Distributed Microcomputers, *Proceedings of the Second ASME Computer Engineering Conference*, San Diego, CA, August, 15-19, 1982, pp. 27-31.
6. S. Harmon, Autonomous Robot Submersibles: The Future of Unmanned Submersibles, *Proceedings of the Second ASME Computer Engineering Conference*, San Diego, CA, August 15-19, 1982, pp. 33-36.
7. J. Miller, A Discrete Adaptive Guidance System for a Roving Vehicle, *Proceedings of the IEEE Conference on Decision and Control*, New Orleans, LA, December 7-9, 1977, pp. 566-575.
8. M. Takano and G. Odawara, Development of New Type of Mobile Robot TO-ROVER, *Proceedings of the Thirteenth International Symposium on Industrial Robots*, Chicago, IL, April 17-21, 1983, pp. 20/81-89.
9. R. McGhee, K. Olson, and R. Briggs, Electronic Coordination of Joint Motions for Terrain-Adaptive Robot Vehicles, SAE Technical Paper Series 800382, SAE Congress, Detroit, MI, February 25-26, 1980.
10. M. Raibert, "Hopping in legged systems: Modeling and simulation for the two-dimensional one-legged case, *IEEE Trans. Syst. Man Cybernet.* SMC-14(3), 451-463 (May/June 1984).
11. J. Vertut, VIRGULE: A Rescue Vehicle of the New Teleoperator Generation, *Proceedings of the Second Conference on Industrial Robot Technology*, Birmingham, U.K., March 17-29, 1974, pp. D3/21-38.
12. M. Russell, Jr., "ODEX-1 the first functionoid," *Unman. Syst.* 2(2), 9-11 (Fall 1983).
13. S. Yerazunis, D. Frederick, and J. Krajewski, Guidance and Control of an Autonomous Rover for Planetary Exploration, *Proceedings of the IEEE Milwaukee Symposium on Automatic Computation and Control*, Milwaukee, WI, April 22-24, 1976, pp. 7-16.
14. B. Carlisle, An Omni-Directional Mobile Robot, in B. Rooks (ed.), *Developments in Robotics 1983*, IFS Publications, Kempston, UK, pp. 79-87, 1983.
15. H. Warnecke and J. Schuler, Mobile Robots: A Solution for the Integration of Transport and Handling Operations, *Proceedings of the Second Conference on Automated Guided Vehicle Systems*, Stuttgart, FRG, June 7-9, 1983, pp. 185-194.
16. M. Larcombe, The Efficient Use of Mini-Computers in Industrial Robot Control, *Proceedings of the Second Conference on Industrial Robot Technology*, Birmingham, UK, March 27-29, 1974, pp. C4-35-42.
17. P. Baylou, G. Bousseau, C. Bouvet, and M. Monsion, Computer Control of an Agricultural Locomotive Robot, *Proceedings of the Second Conference on Automated Guided Vehicle Systems*, Stuttgart, FRG, June 7-9, 1983, pp. 243-249.
18. M. Yachida, T. Ichinose, and S. Tsuji, Model-Guided Monitoring of a Building Environment by a Mobile Robot, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, August 8-12, 1983, pp. 1125-1127.
19. G. Bauzil, M. Briot, and P. Ribes, A Navigation Sub-System Using Ultrasonic Sensors for the Mobile Robot HILARE, *Proceedings of the First Conference on Robot Vision and Sensory Control*, Stratford-Upon-Avon, U.K., April 1-3, 1981, pp. 47-58.
20. M. Julliere, L. Marce, and H. Perrichot, A Guidance System for a Vehicle Which Has to Follow a Memorized Path, *Proceedings of the Second International Conference on Automated Guided Vehicles*, Stuttgart, FRG, June 7-9, 1983, pp. 211-221.
21. H. Moravec, "The Stanford Cart and the CMU Rover," *Proc. IEEE* 71(7), 872-884 (July 1983).
22. J. Crowley, "Navigation of an intelligent mobile robot," *IEEE J. Robot. Autom.* RA-1(1), 31-41 (March 1985).
23. R. Wallace, A. Stentz, C. Thorpe, H. Moravec, W. Whittaker, and T. Kanade, First Results in Robot Road-Following, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 18-23, 1985, pp. 1089-1095.
24. R. Andersson and R. Bajcsy, SCIMR: A Test Bed for Real Time Processing of Sensory Data, *Proceedings of the IEEE Conference on Pattern Recognition and Image Processing*, Las Vegas, NV, June 14-17, 1982, pp. 355-357.
25. C. Witkowski, A Parallel Processor Algorithm for Robot Route Planning, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, August 8-12, 1983, pp. 827-829.
26. Y. Kanayama, Concurrent Programming of Intelligent Robots, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, August 8-12, 1983, pp. 834-838.
27. H. R. Everett, "A second-generation autonomous sentry robot," *Robot. Age*, 29-32 (April 1985).
28. S. Tsugawa, T. Hirose, and T. Yatabe, An Intelligent Vehicle with Obstacle Detection and Navigation Functions, *Proceedings of the IEEE International Conference on Industrial Electronics, Controls and Instrumentation*, Tokyo, Japan, October 22-26, 1984, pp. 303-308.
29. S. Harmon, USMC Ground Surveillance Robot: A Testbed for Autonomous Vehicle Research, *Proceedings of the Fourth UAH/UAB Robotics Conference*, Huntsville, AL, April 24-26, 1984.
30. J. Iijima, Y. Kanayama, and S. Yuta, A Locomotion Control System for Mobile Robots, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC, August 24-28, 1981, pp. 779-784.
31. D. Okhotsimski, V. Gurfinkel, E. Devyanin, and A. Platonov, Integrated Walking Robot, in J. E. Hayes et al. (eds.), *Machine Intelligence, Vol. 9*, Wiley, New York, pp. 313-329, 1979.
32. V. Lapshin, Control of Four-Legged Running Type Robot, *Artificial Intelligence and Information-Control Systems of Robots*, in I. Plander (ed.), Elsevier Science, Amsterdam, The Netherlands, pp. 225-228, 1984.
33. S. Premi and C. Besant, A Review of Various Vehicle Guidance Techniques that Can Be Used by Mobile Robots or AGVS, *Proceedings of the Second International Conference on Automated Guided Vehicle Systems*, Stuttgart, FRG, June 7-9, 1983, pp. 195-209.
34. S. Riter and J. McCoy, "Automatic vehicle location: An overview," *IEEE Trans. Vehic. Technol.* VT-26(1), 7-11 (February 1977).
35. S. Harmon, Knowledge Based Position Location on Mobile Robots, *Proceedings of the Eleventh IEEE Industrial Electronics Society Conference*, San Francisco, CA, November 18-22, 1985.
36. R. Berry, K. Loebbaka, and E. Hall, Sensors for Mobile Robots, *Proceedings of the Third Conference on Robot Vision and Sensory Controls*, Cambridge, MA, November 7-10, 1983, pp. 584-588.
37. A. Elfes and S. N. Talukdar, Distributed Control System for the CMU Rover, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, August 8-12, 1983, pp. 830-833.
38. R. Chatila and J.-P. Laumond, Position Referencing and Consistent World Modeling for Mobile Robots, *Proceedings of the IEEE Conference on Robotics and Automation*, St. Louis, MO, March 25-28, 1985, pp. 138-145.
39. Y. Ichikawa and M. Senoh, A Heuristic Guidance System for Automated Vehicles, *Proceedings of the IEEE Conference on Industrial Electronics, Controls & Instrumentation*, Tokyo, Japan, October, 22-26, 1984, pp. 313-317.



40. C. Zhao, S. Monchard, L. Marce, and M. Julliere, Location of a Vehicle with a Laser Range Finder, *Proceedings of the Third International Conference on Robot Vision and Sensory Controls*, Cambridge, MA, November 7–10, 1983, pp. 82–87.
41. J. Courtney and J. Aggarwal, Robot Guidance Using Computer Vision, *Proceedings of the 1983 IEEE Trends and Applications*, Gaithersburg, MD, May 25–26, 1983, pp. 57–62.
42. I. Fukui, "TV image processing to determine the position of a robot vehicle," *Patt. Recog.* **14**(1–6), 101–109 (1981).
43. K. Kamejima, Y. Ogawa, and Y. Nakano, Perception-Control Architecture in Image Processing for Mobile Robot Navigation System, *Proceedings of the IEEE International Conference Industrial Electronics, Control and Instrumentation*, Tokyo, Japan, October, 22–26, 1984, pp. 52–57.
44. A. Rosenfeld, L. Davis, and A. Waxman, Vision for Autonomous Navigation, *Proceedings of the First IEEE Conference on Artificial Intelligence Applic.*, Denver, CO, December 5–7, 1984, pp. 140–141.
45. B. Lucas and T. Kanade, Optical Navigation by the Method of Differences, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 18–23, 1985, pp. 981–984.
46. K. Drake, E. McVey, and R. Inigo, "Sensing error for a mobile robot using line navigation," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-7**(4), 485–490 (July 1985).
47. D. Guentri and L. Norton-Wayne, Automatic Guidance of Vehicle Using Visual Data, *Proceedings of the Fifth IEEE Conference on Pattern Recognition*, Miami Beach, FL, December 1–4, 1980, pp. 146–149.
48. R. Jarvis, "A perspective on range finding techniques for computer vision," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-5**(2), 122–139 (March 1983).
49. S. Harmon, Coordination between Control and Knowledge Based Systems for Autonomous Vehicle Guidance, *Proceedings of the IEEE Trends and Applications*, Gaithersburg, MD, May 25–26, 1983, pp. 8–11.
50. D. Keirsey, E. Koch, J. McKisson, A. Meystel, and J. Mitchell, Algorithm of Navigation for a Mobile Robot, *Proceedings of the IEEE Conference on Robotics*, Atlanta, GA, March 13–15, 1984, pp. 574–583.
51. V. Pyatkin and V. Sirotenko, "Path planning by robot," *Eng. Cybernet.* **16**(6), 54–59 (November/December 1978).
52. S. Harmon, Comments on Automated Route Planning in Unknown Natural Terrain, *Proceedings of the IEEE Conference on Robotics*, Atlanta, GA, March 1984, pp. 571–573.
53. R. Sobek, A Robot Planning Structure Using Production Rules, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 18–23, 1985, pp. 1103–1105.
54. B. Dobrotin and D. Rennels, An Application of Microprocessors to a Mars Roving Vehicle, *Proceedings of the 1977 Joint Automatic Control Conference*, San Francisco, CA, June 22–24, 1977, pp. 185–196.
55. N. Bogomolov, Yu. Lazutin, and V. Yaroshevshy, The Multiprocessor Control System of a Mobile Robot with Elements of Artificial Intelligence, in I. Plander (ed.), *Artificial Intelligence and Information-Control Systems for Robots*, Elsevier Science, Amsterdam, Netherlands, pp. 91–95, 1984.
56. S. Harmon, D. Gage, W. Aviles, and G. Bianchini, Coordination of Intelligent Subsystems in Complex Robots, *Proceedings of the IEEE Conference on Artificial Intelligence Applications*, Denver, CO, December 5–7, 1984, pp. 64–69.
57. J. Lupo, "Tactical autonomous weapons systems," *Unman. Syst.* **2**(4), 7–9 (Spring 1984).
58. J. Corrado, "Military robots," *Design News*, 45–65 (October 10, 1983).

59. R. Freitas, T. Healy, and J. Long, Advanced Automation for Space Missions, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC, August 24–28, 1981, pp. 803–807.

S. HARMON  
Robot Intelligence International

## ROSIE

A software tool to facilitate the building of expert systems (qv), ROSIE is a general-purpose, rule-based, procedure-oriented system. The system, when provided with a knowledge base, will become an expert system for the domain embodied by the knowledge base. It was developed around 1981 at Rand Corporation (see J. Fain, et al., *The Rosie Language Reference Manual*, Technical Note N-1647-ARPA, Rand Corporation 1981, and F. Hayes-Roth, et al., *Rationale and Motivation for Rosie*, Technical Note N-1648-ARPA, Rand Corporation, 1981).

K. S. ARORA  
SUNY at Buffalo

## RULE-BASED SYSTEMS

Rule-based systems (RBSs) constitute the best means available today for codifying the problem-solving know-how of human expert. Empirically, it seems that experts can express most of their problem-solving techniques simply as a set of situation–action rules. This makes RBSs the method of choice for building knowledge-intensive expert systems. Although many different techniques have emerged to organize collections of rules into automated experts, RBSs share many key properties:

An RBS incorporates practical human knowledge expressed in terms of conditional if–then rules.

An RBS grows in skill as its collection of rules is incrementally expanded.

The RBS can solve a wide range of possibly complex problems by selecting relevant rules and combining their results in appropriate ways.

The RBS determines dynamically the best rules to execute.

The RBS explains its conclusions by retracing its actual line of reasoning and translating to English the logic of each rule employed (see Explanation).

Rule-based systems address the need to capture, represent, store, distribute, reason about, and apply human knowledge electronically. Within the current state of the art they provide a practical means of building automated experts in application areas where job excellence requires consistent reasoning and rewards practical experience. Table 1 lists some application areas addressed by current RBS technology.

RBSs take their name from the way they represent knowledge about plausible inferences and preferred problem-solving tactics. Some of them are listed in Table 2. Typically, they represent both sorts of know-how in terms of conditional rules. The examples below illustrate rules taken from a variety of applications, employing varying syntactic conventions.

**Table 1. Rule-Based System Applications**

Problem	System Functions
Equipment maintenance	Diagnose faults and recommend repairs
Component selection	Elicit requirements and match parts catalog
Computer operation	Analyze requirements; select and operate software
Product configuration	Elicit preferences and identify parts that satisfy constraints
Troubleshooting	Analyze situation, suggest treatments, and prescribe preventative measures
Process control	Spot problematic data and remedy irregularities
Quality assurance	Assess task, propose practices, and enforce requirements

**Table 2. Some Rule-Based Expert Systems**

ACTOR	MAC
AGE	Meta-DENDRAL
AGE-PUFF	MRS
AMORD	MYCIN
BB-1	NEOMYCIN
BIP	NESTOR
CASNET	NUDGE
CLOT	OCEAN
CRYSLIS	OPS
DART	PLANNER
DENDRAL	PROLOG
Diagnosis I	PROSPECTOR
Diagnosis II	PUFF
EMYCIN	RITA
EXPERT	ROGET
FRL	ROSIE
HASP	S.1
HEARSAY II	SACON
INTERNIST-1	SAIL
IRIS	SMALLTALK
KRL	SOPHIE
LITHO	TIERESIAS
M.1	XCON

**Example 1.** Automotive troubleshooting rules represented in an S.1 program. As cars incorporate more electronic subsystems, they become more difficult for average technicians to repair. S.1 provides a structural framework for organizing and applying thousands of rules. General Motors plans to aid its service technicians with several large-scale RBSs.

Rule408:

C is a car.

If: the pattern observed by attaching an oscilloscope to the charging circuit of the car C is fluctuating arches, and  
the alternator of the car C responds properly to different loads,  
then: there is strongly suggestive evidence (.9) that the cause of the problem with the car C is voltage-regulator bad.

Rule428:

C is a car.

If: the pattern obtained by attaching an oscilloscope to the charging circuit of the car C is straight.line, and  
the result pulling out the field connector is no.flash, and  
the field connector does not have a voltage, and  
the input of the voltage regulator does not have a voltage, and  
the dashboard's lights do not glow when their ground circuit is completed, and  
the fusable link is getting voltage, and  
the fusable link is not conducting power,  
then: it is definite (1.0) that the cause of the problem with the car C is fusable.link.bad.

**Example 2.** Legal heuristics for product liability represented in a ROSIE (qv) program. Each rule expresses an independent chunk of know-how. ROSIE provides a stylized English-like syntax for expressing conditions and actions.

If the plaintiff did receive an eye injury  
and there was just one eye that was injured  
and the treatment for the eye did require surgery  
and the recovery from the injury was almost complete  
and visual acuity was slightly reduced by the injury  
and the condition is fixed,  
increase the injury trauma factor by \$10,000.  
If the plaintiff's injury did cause  
(a temporary disability of an important function)  
and the plaintiff's doctors were not certain about  
the disability being temporary  
and the plaintiff's recovery was almost complete  
and the condition is fixed,  
increase the fear factor by \$1000 per day.  
If the plaintiff did not wear glasses before the injury  
and the plaintiff's injury does require  
(the plaintiff to wear glasses),  
increase the faculty loss factor by \$1500  
and increase the inconvenience factor by \$1500.

Rule-based systems may be defined as modularized know-how systems. Know-how refers to practical problem-solving knowledge. It consists of a variety of kinds of information, including inferences that follow from observations; abstractions, generalizations, and categorizations of given data; necessary and sufficient conditions for achieving some goal; suggested places to look for information one might need; preferred

strategies for eliminating uncertainty or minimizing other risks; likely consequences of hypothetical situations; and probable causes of symptoms.

Today's RBS technology provides the first practical methodology and notation for developing systems capable of knowledge-intensive automated performance. Although AI researchers have developed several alternatives, only the RBS approach consistently produces expert problem solvers. This reflects a feature of the current state of the art in automatic reasoning (qv), namely that RBSs can incorporate directly rules that emulate the effective special-case reasoning characteristic of highly experienced professionals. On the other hand, general-purpose deductive schemes lack the efficiency required to solve complex practical tasks. Because each rule approximates an independent nugget of know-how, RBS development has two key characteristic features: The systems improve performance incrementally as system builders refine the existing and add new knowledge and the ability of the systems to explain their reasoning makes their logic practically transparent, which meets a widely recognized need for understandability of computer systems.

By incorporating know-how, acquired in an incremental and transparent manner, RBSs open up key computing applications not readily addressable by alternative techniques. These include tasks where the demands for quality performance by humans exceed the supply. Such demands may arise from a variety of causes, including: workers in the same job may differ significantly in job abilities; expert job performance produces significantly better results than average; and conventional means of training and automation prove inadequate to produce expert performance. Automating expertise in specialized tasks generally requires a few hundred to a few thousand heuristic rules. With existing technology, this makes good economic sense in hundreds of application areas.

In spite of its relevance and potential, RBS technology has shortcomings. It represents a new technology that will improve for many years. Applications of the technology today must be assessed carefully for their feasibility and deployability. Only a small fraction of all the potential uses of this technology can be addressed with off-the-shelf technology today. Most applications today require some customized engineering.

Nevertheless, RBSs constitute the best means available today for building expert systems that incorporate large amounts of judgmental, heuristic, experiential know-how. Widespread utilization of this technology has just begun, owing to the recent release of numerous commercial products recently. Informal surveys indicate that approximately 50% of the Fortune-500 companies are investigating this technology, and about 10% of them now have applications under development.

RBSs embody an operating concept that differs radically from von Neumann architectures. In this concept intelligent problem-solving means an iterative cycle of identifying from experience those heuristic rules that bear on a problem at hand and applying one of those rules to solve or simplify the problem. The technology for building RBSs supports this cycle by providing a dynamic working memory for partial results, a device to identify relevant rules, and selective means for applying desirable rules. Many people conjecture that human problem-solving activity follows the RBS model. Whether that proves true, human experts generally find it easy to express methods for solving problems in their application areas using a rule formation.

The technology for building RBSs has progressed significantly in the last 10 years. During that time many people have analyzed the technology and assessed its relevance for a variety of tasks. Today, there are news reports, such as:

Company X improves speech-synthesis systems using AI technology to incorporate hundreds of heuristic rules.

Company Y saves millions of dollars by employing a rule-based expert system that checks incoming orders and screens sales errors.

Company Z deploys a rule-based loan advisor to standardize loan evaluation throughout all its offices.

Today rule-oriented components are becoming central in many advanced computing applications. This entry surveys the state of RBS technology. The goal is to help readers determine whether this technology can contribute significantly in their own areas of interest. The following sections present an overview of RBS techniques, discuss the RBS niche in the larger field of computing, look at the structure of a rule and the architecture of an entire RBS, review the conceptual and historical development of RBSs, and evaluate the current state of the art.

### The RBS in Overview

A simplified form of the rule-based system consists of storage and processing elements, often loosely labeled as the knowledge base plus inference engine (see Fig. 1) The basic cycle of a rule-based system consists of select and execute phases. In the select phase the system determines which rules can apply and chooses one in particular to execute. In the execute phase the system interprets the selected rule to draw inferences that alters the system's dynamic memory. System storage includes components for long-term static data and short-term dynamic data. The long-term store, called the knowledge base, contains rules and facts. Rules specify actions the system should initiate when certain triggering conditions occur. The conditions define important patterns of data that can arise within the working memory. The system represents data in terms of relations, propositions, or equivalent logical expressions. Facts define static, true propositions. In contrast with conventional DP systems, the basic RBS distributes its logic over numerous independent condition-action rules, monitors dynamic results for triggering patterns of data, determines its sequential behavior by selecting its next activity from a set of candidate triggered rules, and stores its intermediate results exclusively in a global working memory.

Roughly speaking, an RBS consists of a knowledge base plus an inference engine. A knowledge base consists of rules and facts. Rules always express a conditional, with an antecedent and a consequent component. The interpretation of a rule is that if the antecedent condition can be satisfied, the consequent can be too. When the consequent defines an action, the effect of satisfying the antecedent is to schedule the action for execution. When the consequent defines a conclusion, the effect is to infer the conclusion.

Because the behavior of the RBS derives from this simple regimen, given particular problem-solving data, the rules completely specify the actual behavior of the system. As a means of specifying behavior, rules perform a variety of distinctive functions. First, the rules decompose the overall state-

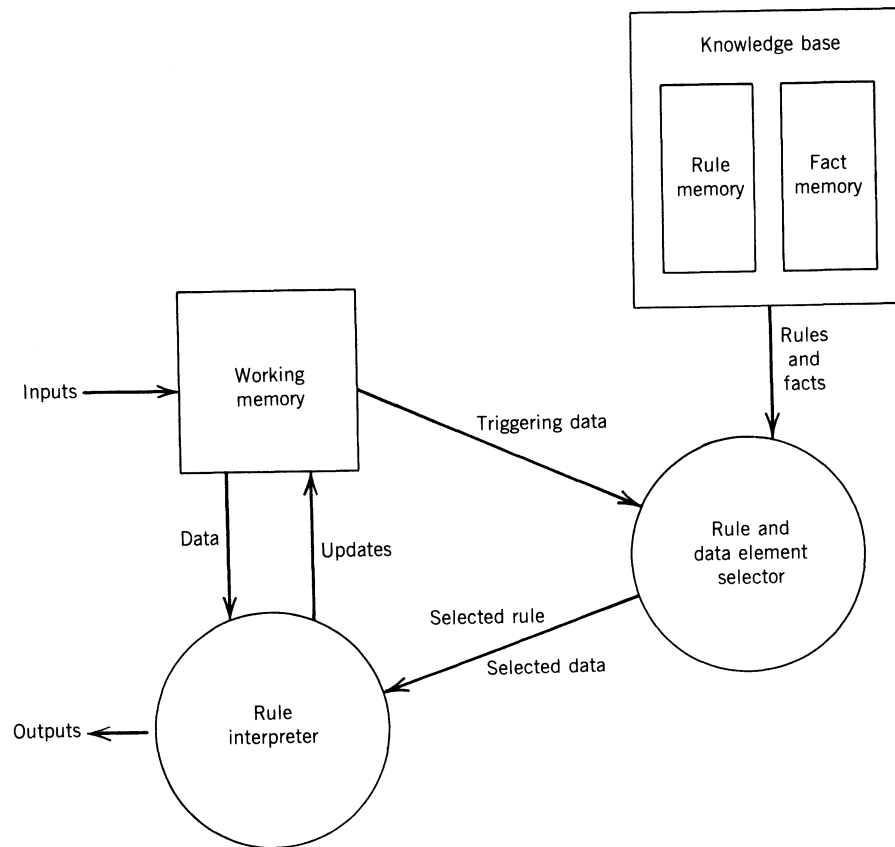


Figure 1. Basic features of rule-based systems.

transition behavior into parallel actions that respond independently to distinct properties of a system state. This in turn simplifies the tasks of auditing and explaining system behavior. Every result is traceable to its antecedent data and intermediate rule-based inferences. Second, because rules can express logical relationships (conditionals) and definitional equivalences, they provide a means to simulate deduction and reasoning. Third, when the rules translate low-level signal data into higher level pattern classes, they can simulate human perception. Finally, by using conditional rules to express rules of thumb, often called heuristics (qv), the RBS simulates subjective decision making.

Several major ways have emerged for organizing RBSs. For example, rules may express deductive knowledge, such as logical relationships. This knowledge can support inference, verifications, or evaluation tasks. On the other hand, rules may express goal-oriented knowledge, which an RBS applies to seek a problem solution and cites to justify its own goal-seeking behavior. Finally, rules may express causal relationships, which an RBS can use to answer "what if" questions or to determine possible causes for specified events (see Reasoning, causal).

An RBS can only solve problems when it incorporates effective rules. The rules use symbolic descriptions to characterize relevant situations and corresponding actions. The language employed for these descriptions imposes a conceptual framework on the problem and its solution. The rules may be precise or gross; the intermediate partial solutions may be abstract or detailed. Efforts to solve the problem may proceed top-down, bottom-up, or in some other way (see Processing, bottom-up

and top-down). The meaning, importance, and contribution of each rule depends on its effectiveness as a contributor within the entire set of rules available for solving a problem.

Facts constitute the other kind of data in a knowledge base. Facts express assertions about properties, relations, propositions, etc. Where each RBS gives an imperative interpretation to rule-based knowledge, the RBS typically views the fact as static and inactive. Implicitly, a fact is silent regarding the pragmatic value and dynamic utilization of its knowledge. Thus, although in many cases facts and rules may be logically interchangeable, RBS distinguish facts and rules for performance reasons. Rules specify in a conditional form potential inferences that the RBS must consider as a basis for its next action.

In addition to its static memory for facts and rules, the RBS employs a working memory for storing temporary assertions. These assertions record earlier rule-based inferences. One can interpret the contents of working memory as problem-solving state information. Ordinarily, the data in working memory adhere to the syntactic convention of facts. The temporary assertions thus correspond to dynamic facts.

The computing environment for interpreting rules consists of the current facts and the inference engine itself. Together these provide a context for interpreting the current state, understanding what the rules mean, and applying relevant rules appropriately. In the examples given in the first section one can detect hints of this implicit frame of reference. The legal rules specify changes to make to various "factors," and the auto-repair rules assert conclusions about causes of problems. These rules are not universally valid. They each depend on

many unstated assumptions that characterizes the implicit frame of reference where they express valid relationships. The validity of these rules depends critically on their being interpreted in the right context. Generally, RBSs cannot obviate all concerns of conventional computer programming—such as representation of state, control of sequencing, and variable scoping—because someone has to ensure that as a computer program the RBS applies rules appropriately in their meaningful contexts. Many people mistakenly assume that RBSs turn unstructured heaps of universally valid, independent rules into effective problem solvers. That is a serious misinterpretation of the current state of technology. So, the rule writer must consider the rule-interpretation environment when the rule is written. By employing knowledge of this context, many RBSs can translate a rule or its applications to produce explanations using convenient notations or excellent English.

The basic function of the RBS is to produce results. The primary output may be a problem solution, an answer to a question, or an analysis of data. Whatever the case, the RBS will employ several key processes in determining its overall activity. A “world” manager maintains information in working memory. A built-in control procedure will define the basic, high-level loop. And if the built-in control provides for programmable specialized control, an additional process will manage branching to and returning from special control blocks.

### The RBS Niche in Computing

RBSs combine a variety of techniques and address numerous shortcomings apparent in conventional technology. Some of the needs RBSs address are

- prescribing how complex programs should behave,
- adapting rapidly to requirements changes arising during development,
- involving users and experts in specifying program operation,
- developing computer-based competence experimentally, and
- capturing and distributing the expertise needed to exploit existing computer capabilities.

The features of RBSs that address these problems include

- modularity of know-how;
- knowledge bases for storing rules and facts that directly determine decisions;
- incremental development, with steady performance improvements;
- explanation of results, lines of reasoning, and questions asked;
- intelligibility of encoded beliefs and problem-solving techniques; and
- dynamic assembly of inference chains, within the context of a built-in control procedure.

Given the importance of the problems for which these techniques seem appropriate, one should expect the RBS niche to expand over time. Rules as a basis for representing knowledge can provide the right solution to some of these problems.

### The Rule as Object

Rules may contain much information beyond their simple conditional if-then component (see Fig. 2). Whereas the antecedent and consequent of a rule specify data sufficient for inferring a conclusion or performing another action, other parts of a rule serve additional important roles. Many large RBSs benefit from hierarchical structuring, in which each rule may belong to one or more higher order collections. These collections, called rulesets, aggregate and differentiate rules according to their function within the system. An RBS may ignore all rules in rulesets momentarily deemed irrelevant to a problem. Data about the rule, such as who wrote it and when, can support testing, evaluation, explanation, and maintenance. Typically, each rule exists in several alternative representations derived by translation. One machine-oriented translation serves the need for high performance at run time; one human-oriented form uses English to support publication and explanation, and another exploits terseness to improve reading and editing. Yet other facets of the rule structure determine how the inference engine should treat the rule. The system may need to trace rule evaluations and applications, justify the rule's relevance, or selectively ignore the rule under various conditions.

The rule is spoken of as a relatively independent piece or chunk of know-how. Psychologists, for some time, have emphasized the subjective reality of chunks. Chunks correspond to the elementary patterns people perceive and manipulate in thinking. They differ from person to person. They reflect the learned, appropriate, effective distinctions in each person's skill areas. A rule corresponds to a chunk of problem-solving know-how.

As used by most RBSs, the rule specifies analytic problem-solving knowledge. The rule is a datum employed by an inference engine to infer a solution to its goal problem. Thus, when the rule writer expresses know-how in rule format, he is offering one possible path to reduce a goal to subgoals, or draw a plausible inference from plausible data, or transform an expression. This information about the rule typically comprises its familiar if-then components. However, as the number of rules in an RBS grows, maintainers need additional assistance to extend and maintain the knowledge base. For this reason, many additional facets or attributes are introduced. These generally represent data about the rule's analytic knowledge and its preferred manner of use.

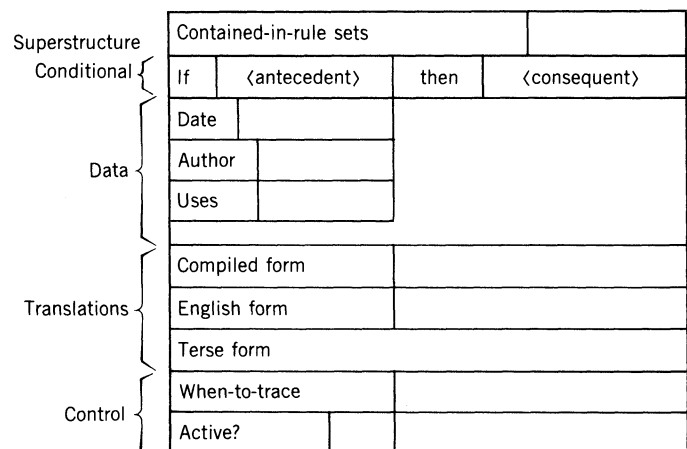


Figure 2. Rule components support multiple functions.

### RBS Architecture

An RBS generally is a complete computing system: It takes inputs, uses memory and processing, and produces outputs. The key elements of RBS technology include rules, interpreters, translations, and explanations.

From the architectural perspective, rules constitute data. Generally these data conform to highly specialized grammars capable of using symbolic expressions to define conditions and actions. Current systems differ primarily in the generality and notational convenience their symbolologies support.

The rule interpreter matches a rule component to working-memory data. Generally this requires pattern matching (qv) that finds constants in working memory that match identical constants or unbound variables in rule patterns. Existing systems differ primarily in the methods they use to simplify rule definition and pattern matching. The action of the rule is produced by another part of the rule interpreter. Actions generally consist of two sorts: changes to working memory or external actions such as I/O.

Some RBSs employ multiple representations of rules, such as one for data entry, another for interpretation, and another for explanations. Typically, all rules are maintained in one preferred representation and translated as needed for other purposes.

The hallmark of RBSs has been their ability to explain their conclusions. Explanations (qv) have been generated by translating to English the rules used in reaching the result of interest. This requires maintaining a history of working memory changes and their causes. This history can be searched as needed for explanation.

RBSs have been organized in a variety of ways. An RBS organization consists of a set of decisions about what meaning to give rules and how and when to interpret them. Two organizations are most common: stimulus-driven, often called forward-chaining, and goal-directed, usually called backchaining. In the former case a rule is triggered when changes in working-memory data produce a situation that matches its antecedent component. Some RBSs allow rules to fire repeatedly as long as the working data still match the rule, but most process a specific working-memory data configuration only once for each rule. In backchaining the RBS begins with a goal and successively examines rules whose consequent components match it. One at a time these candidate rules are considered. From each such rule whose applicability remains plausible, the unmet conditions of the antecedent are extracted. Each such condition, in turn, is defined to be a new goal. And the backchaining control procedure shifts attention toward the new goal, recursively. This effort terminates whenever the top goal has been reduced to a set of satisfied subgoals.

From the point of view of computer architecture, two kernel facilities distinguish RBSs from conventional systems. First, the RBS makes heavy use of pattern matching between rule components and working memory. Second, RBSs must quickly identify rules that become relevant as working memory changes. This requires a means of accessing rules by pattern-matched values. Most RBSs support these requirements today in software, although some current hardware efforts aim at improving performance on these tasks.

The simple model of the RBS, consisting merely of a knowledge base and inference engine, undergoes substantial modification as the RBS advances in complexity (see Fig. 3). Two primary objectives motivate these elaborations: clarity of the

knowledge and run-time performance. Knowledge clarity meets many of the key requirements for RBS development. These include expressibility of know-how by experts; intelligibility of the knowledge and related reasoning to experts, their peers and users; and modifiability and extensibility. Many of the differentiated features evident in the advanced system address the concern for enhanced knowledge clarity. These include the elaboration of a multidimensional working memory and the separation among rules, metarules, and control procedures. The goal of performance also motivates many of the embellishments the advanced system incorporates. A rule compiler converts the triggering data conditions into a data-flow network that optimizes the computing required to identify executable rules. The advanced system introduces three additional data sources to aid in the selection of the one rule to execute next. Higher level rules called metarules express preferences and priorities that differentially favor specific candidates (see Meta-knowledge, -rules, and -reasoning). The prioritized list of rules awaiting execution constitutes the agenda (see Agenda-based systems). The process examines the agenda of waiting rules and whatever specialized control procedures the system includes. The scheduler selects for execution next either a new procedure, a procedure continuation, or the action of a high-priority rule.

### Role in Evolutionary System Development

RBSs have proved invaluable as a practical means for evolving poorly understood knowledge. Although today's RBSs cannot substitute for the full range of mature DP application-building technology, they offer a unique advantage missing from the conventional tool-kit. We should anticipate that as the technology matures, the essential ingredients of RBSs will migrate into DP technology as a basis for rapid prototyping, improving extensibility, and enhancing software maintenance and support.

### Relationship to Search

The focus on knowledge in applied AI systems represents a reaction to the abortive attempts to solve important problems using general-purpose or weak methods. As the importance of knowledge became clear, many AI researchers became knowledge engineers. They emphasized picking high-value problems with symbolic solutions, identifying corresponding human experts, and debriefing the experts to find out what they knew.

In many cases the power of expertise corresponded to hints and tricks for obviating work that a less experienced person might need to perform. Often, this meant reducing what would be a big search space for a general problem-solving program to a small search space for a specialized, knowledge-intensive program. So although some RBSs perform search (qv), they generally incorporate a representation of the problem and chunks of the solution that greatly simplify the task. Search is a last resort for problemsolvers. Most rule-based systems perform little or no search.

### Relationship to Programming

As suggested earlier, today's RBS technology requires the rule writer to consider and understand the organization and operation of the target RBS. That means writing rules is a special



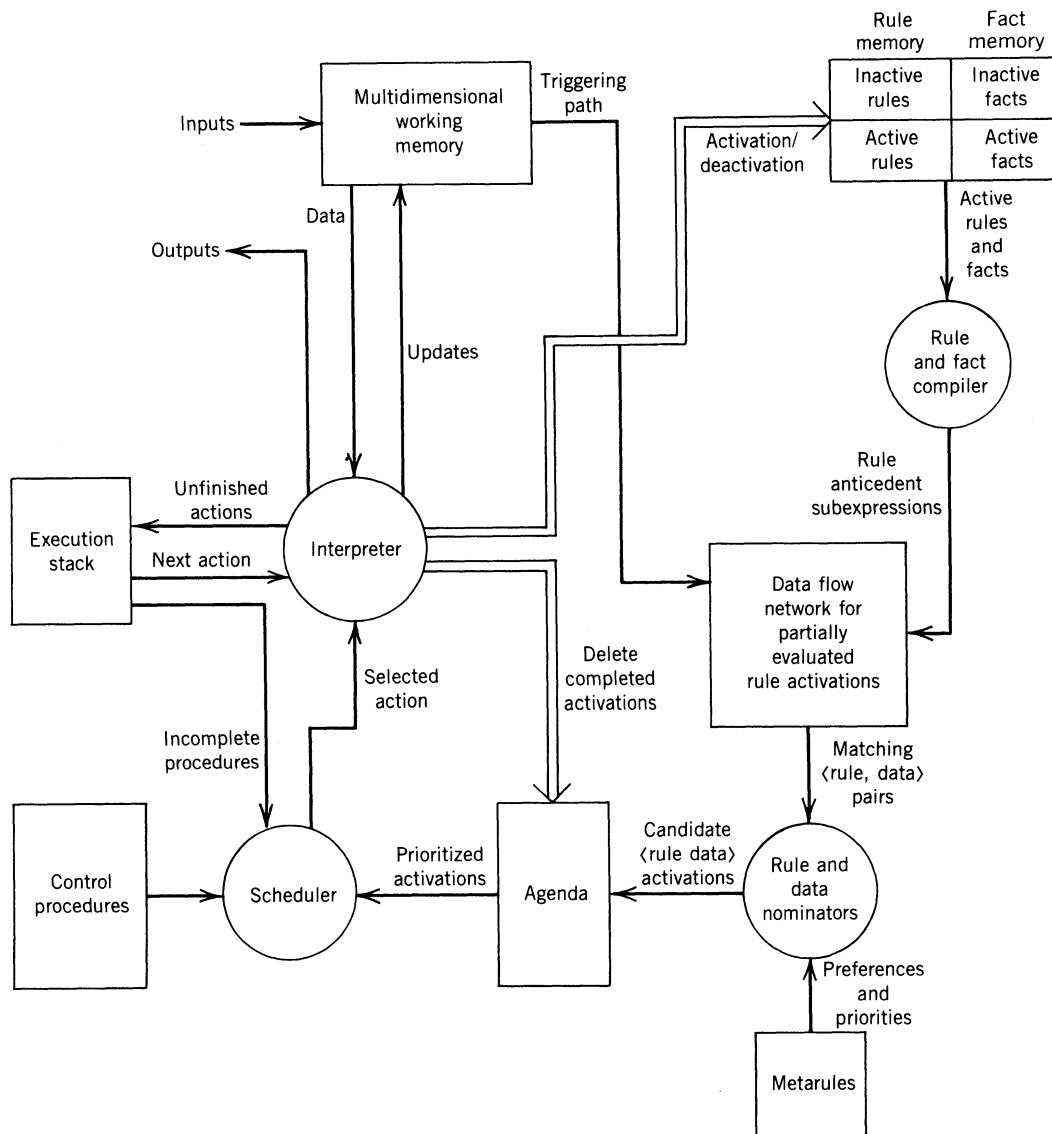


Figure 3. Representative advanced rule-based system.

kind of programming. Like conventional programming, effective rule programming requires mental modeling of state changes, syntactic and semantic checking of rule conditions and execution effects, and heuristic methods to validate and verify the proposed system. In contrast with conventional (procedural) programming, however, rule-based programming requires the author to think more analytically than procedurally. Most programmers have some difficulty with this for a few weeks. Conventional programming requires the programmer first to appreciate the relevant goals and heuristic methods and then to implement a corresponding customized problem-solving program. Rule-based programming, in contrast, requires the programmer first to understand the general method of rule-based problem solving and then to express the current problem description and related heuristic methods in a form consistent with the available knowledge base and inference engine. This requires different skills. Beyond what is normally required, the rule programmer must formulate explicitly the heuristics and problem features. However, the RBS automates nearly everything else required for solving the problem.

One should anticipate that the complimentary strengths of conventional programming and RBSs will motivate research efforts to marry the two technologies so that applications can exploit the advantages of both.

### Conceptual Evolution of RBS

Rule-based systems incorporate many ideas derived from both theory and experience. Figure 4 depicts the evolution of rule-based systems technology resulting from efforts to apply general concepts that originated in the academic disciplines of psychology and computing theory. The discipline of AI does not appear explicitly in the figure because the entire circle depicts a major portion of the field.

The figure portrays the field's evolution in the spiral form of a nautilus, where each new development phase derives from those just prior and rests on previous generations of related developments. Many different paths through the spiral make interesting histories. By following the spiral clockwise, you retrace successive cycles of concurrent activities in the four sectors. By traversing a radial spoke outward from the center,

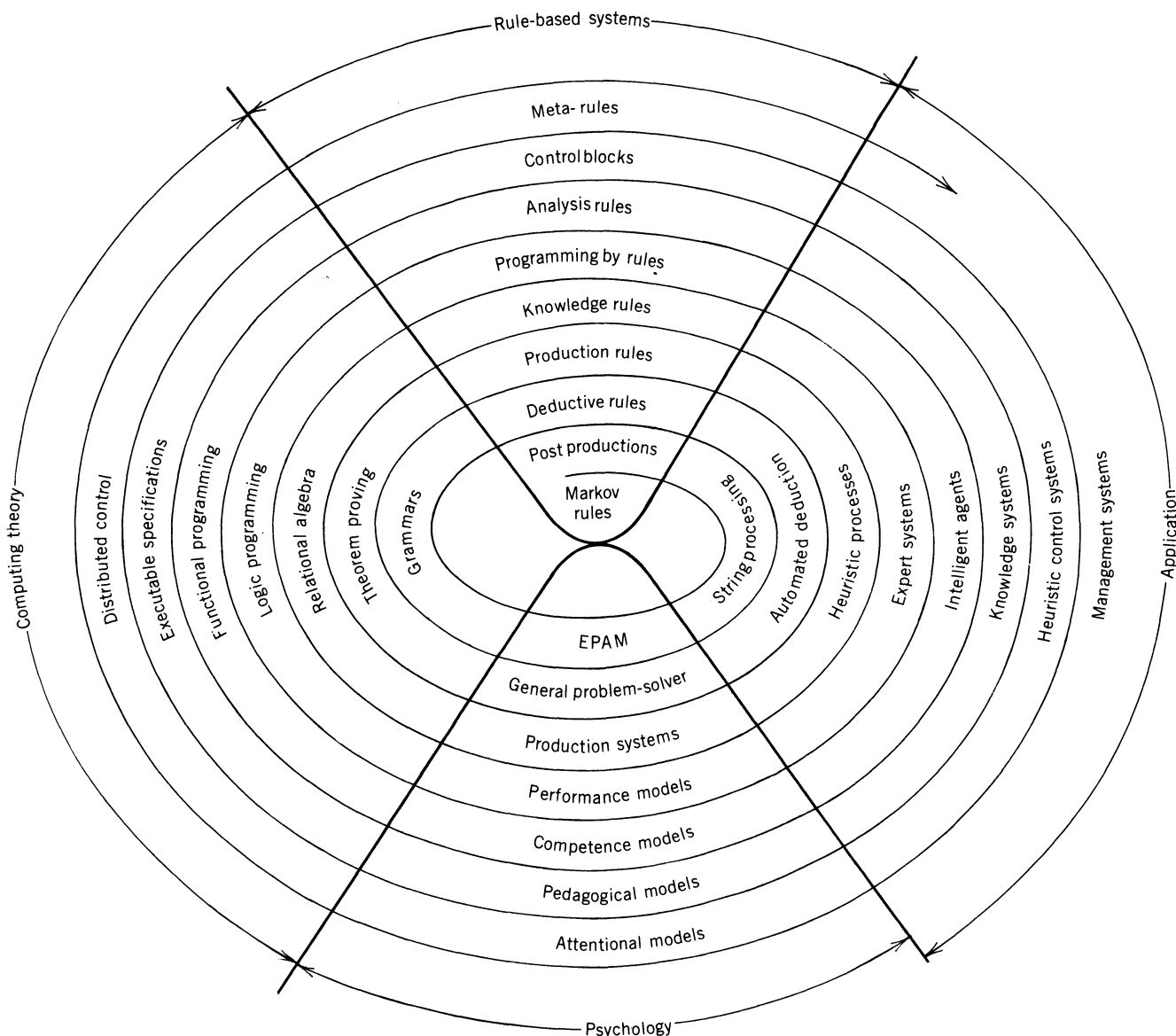


Figure 4. Evolution of rule-based system concepts.

you pass quickly over the successive changes in concept and Zeitgeist within a discipline.

Following a ray within the RBS sector, the major developments in RBS technology are retraced.

At the outset Markov rules provided a simple technique for defining stochastic processes via probabilistic rules that mapped any current state into its possible successors. Post productions showed that machines following simple string-matching condition-action rules could perform all computations. Subsequently, theorem provers emerged that automated deduction. The dual goals of performing human problem-solving tasks and avoiding some gross inefficiencies of general-purpose deduction led to condition-action production-rule systems. These often emphasized low-level and detailed activities so they soon gave way to knowledge rules. Knowledge rules embody chunks of expert know-how. After the initial success of knowledge rules for expert systems, many people began developing general-purpose rule-based programming systems. Often a single rule in these systems combined analytical knowledge about the problem domain and control knowl-

edge about ways to achieve problem-solving efficiency. Subsequently these forms were distinguished. Control blocks expressed imperative knowledge in a procedural form, and metarules emerged to represent other forms of knowledge about knowledge in a rule-based format.

The cognate areas have both supported and adapted to the developments in RBS technology. Developments in computing theory have provided a foundation in automata, grammars, theorem proving, relational algebra, applicative programming styles, executable specifications, and distributed control. Developments in psychology have evolved from extremely general and simple views of intelligent functions as in Markov models, EPAM, and GPS to successively more knowledge-intensive and elaborated views of cognition. These later conceptual generations distinguish what is known from how it can be applied, taught, or made more efficient. The primary foci of applications have shifted over time from general-purpose simulation, string processing, and automated deduction to address more pressing, higher value, and knowledge-intensive concerns. Successively, these tasks have reflected needs for heu-

ristic solutions, specialized expert systems for problem solving, autonomous intelligent agents, knowledge systems for storing and distributing in electronic forms large quantities of institutional knowledge, and heuristic systems for adaptive control and other management tasks.

The RBS of today incorporates numerous influences and has engendered many related technology developments. The essence of the RBS derives from the production-system model used in automata theory and psychology. The basic model was the stimulus-response (S-R) association presumed by some to underlie all animal behavior. In a similar manner theorists sometimes have found it convenient to describe all computational behavior in terms of state-transition tables that define rules for moving between states. An early model of perception called PANDEMONIUM (qv) viewed human-signal-interpretation activity in terms of the actions of independent pattern-action modules called "demons" (qv).

Many researchers have gravitated toward rule-based representations of knowledge for two other reasons. First, rules seem a natural way to express the situation-action heuristics evident in experts' thinking-aloud problem-solving protocols. Second, perhaps more than for any other form of knowledge, researchers have developed learning procedures capable of inferring rules from experience. In addition, RBSs often can accept and assimilate a newly learned rule merely by incorporating it into the knowledge base.

Specialized RBS architectures have evolved to address different target applications. Each specialty seems to benefit from slightly different emphases. Today, special formalisms and supporting systems have been developed in each of these areas: rule-based programming systems; rule-based signal-understanding systems; rule-based cognitive simulations; teachable and learnable rule-based systems; systems for learning rules; and systems for building commercial rule-based expert systems.

By now the key ideas of RBSs have infected many other areas of computing. Chief among these are rule-based subsystems for communications architectures; rule formalisms for representing military doctrine, standard policies, and historical precedents; rules for deduction and programming as rule-based controlled deduction; macrorules in the form of pattern-directed modules for distributed architectures and systems of cooperating experts; metarules for heuristic adaptive control of resource-limited systems; and rules as a basis for enforcing constraints. Although many of these new uses are experimental today, many should mature in the years ahead.

### Technology Evolution of RBSs

The technology of RBSs has incorporated many ideas from diverse sources. A brief and highly simplified recounting of this development follows. This account focuses on the principal developments in computer science that have most advanced the RBS field.

The story begins with decision tables and compilers. This technology emerged about 20 years ago. It provides a representation of decision logic for transaction processing and report generation. A decision-table entry defines a condition-action rule. Rules execute sequentially upon the current input data. The context effects are immediate because there are no working data. The only knowledge-base entries are the rules, which must represent simple Boolean conditions. Large rule tables

prove quite complex; the rigid order of rule evaluation often proves unsatisfactory; an inability to describe complex symbolic patterns and to combine intermediate results dynamically limits the range of applications.

Early AI problem-solving languages, such as PLANNER from MIT, provided means for representing rules within the context of programmable theorem provers. Workers at Carnegie-Mellon (CMU) were the first to build RBSs with thousands of rules and to develop efficient compilers and translators. One RBS initiated at CMU, called XCON, became the first to earn a multimillion-dollar profit, when it was used to eliminate errors in Digital Equipment Corporation VAX orders. The general rule-based programming system called OPS, used for XCON, has been used for several other RBS applications.

Workers at Stanford developed the MYCIN family of RBSs. MYCIN as the first to achieve acceptance by experts, perform expert-level subjective reasoning with uncertain data and knowledge, and explain its reasoning in English. A similar system called PROSPECTOR, developed by SRI, automated knowledge of mineral deposits and is retrospectively credited with identifying a major source for at least one mine operation. (It coincidentally identified the mine source with several human prospectors.) Subsequent work on TEIRESIAS and MRS emphasized metarules for expressing explicit knowledge about control.

Systems at Stanford and CMU for reasoning about signal data have evolved to handle larger macrorules called specialists, knowledge sources, or pattern-directed modules. These systems include HEARSAY-II, HASP, AGE, and BB-1. These systems often pack a great deal of knowledge into a single module. Each module has a condition and an action, and the overall system behaves like the RBSs we have considered. These systems differ from the more typical RBS by using local memory in its computations. In this regard these systems have much in common with object-oriented architectures (such as Smalltalk or Intel's S-432).

PROLOG was the first general-purpose logic-based programming language. It is an RBS that uses stored facts and rules to deduce solutions to goal patterns. It was designed for theorem proving but has proved attractive for a wider range of AI tasks.

The RITA and ROSIE systems, developed at RAND, advanced the concept of using RBS methods for conventional programming. These programming systems blend a rule-based representation for programs with flexible I/O. These capabilities make these research languages very attractive for building automated intelligent assistants for computer-based communication tasks.

The programming system M.1, developed at Teknowledge, incorporates techniques for tolerating uncertainty and combining evidence within a general-purpose rule-based programming system that operates on an IBM personal computer. M.1 marries the rule-based programming capabilities of PROLOG, RITA, and ROSIE to the evidence-combining capabilities of MYCIN. This combination, plus the ability to run on personal computers, has made M.1 the most popular tool for building small-to-medium expert systems. Several other vendors offer tools with some of these features.

The expert-system building tool S.1, also developed at Teknowledge, advanced previous RBS technology by differentiating representations for analytic and imperative knowledge. S.1 employs rules for analytic knowledge and procedural

control blocks for imperative knowledge. It provides a built-in backchaining control mechanism with points of escape for user-supplied control blocks. By separating these two forms of knowledge, users create more intelligible knowledge bases that permit much improved automated explanations. The following example illustrates a control block that defines an approach to organizing rule-based reasoning to diagnose a car. It would employ the earlier car-diagnose rules shown in Example 3.

**Example 3.** A control block expressing procedural know-how in a sophisticated RBS. S.1 provides a procedural syntax for expressing imperative knowledge by distinguishing control blocks and rules; the sophisticated RBS enhances intelligibility and produces improved explanations of its lines of reasoning.

#### High-level.problem-solving.approach:

In order to diagnose and repair a car, follow this procedure:

C is a car.

Display the following:

Welcome to the car-charging diagnosis and repair advisor.

Find out about a car called C.

Determine the initial symptoms of the car C.

Determine the cause of the problem with the car C.

Determine the recommendations for fixing the problem with the car C.

Show the recommendations to fix the problem with the car C.

#### Implementation and Availability

A partial list of supported tools available for building RBSs appears in Table 3.

Current efforts in RBS-related R&D, within, e.g., the DOD-sponsored Strategic Computing Initiative focus on the following objectives: increase the size of practical rule bases to 10,000 rules or more; increase the speed by two orders of magnitude or more; broaden the set of inference techniques used by the interpreters; improve the methods for reasoning with uncertainty; simplify the requirements for creating and extending knowledge bases; and exploit parallel computing in RBS execution.

#### Assessment: Reputation versus Reality

Because RBSs have played an important role in demonstrating the importance and practicality of knowledge systems,

they have received much attention. The reputation of the RBS should now be clarified. The RBS is not a panacea either for DP or AI problems. It represents a technology of broad and important applicability. RBSs differ from conventional programs in several important ways. Compared to many of the techniques of conventional data processing, the younger RBS technology possesses distinct advantages that will advance and improve for years to come.

Rule-based components are becoming standard in advanced applications today. DEC and NCR have incorporated this technology into their XCON and OCEAN order-entry and configuration systems. GM has undertaken several rule-based expert systems for manufacturing and service functions. Numerous aggressive development programs underway throughout the world, including the Japanese Fifth Generation, the British Alvey, the U.S. Strategic Computing, and MCC programs, all aim to improve greatly the performance and generality of this technology over the next 5–10 years.

RBSs have attracted attention because they have important strengths:

RBSs address a largely unmet area of opportunity by representing problem-solving know-how in a manner suitable for application by computers;

RBSs modularize chunks of knowledge;

they support incremental development;

they make decision making more intelligible and explainable;

specialized RBS architectures have emerged that constrain and simplify application methods;

recent advances in RBS technology distinguish imperative and analytic know-how but integrate both to produce more effective, cogent, and maintainable knowledge bases;

rule-based reasoning can provide a conceptual basis for the analytic formulation of imperative know-how;

RBSs provide a fertile framework for conceptualizing computation in general; and

RBSs open new opportunities by providing a non-von Neumann schema for computing systems that can exploit parallelism.

However, RBSs lack several qualities that would make them more suitable as a general computing approach. What are lacking are a precise analytic foundation for the problems solvable by RBSs, a suitable verification methodology or a technique to test the consistency and completeness of a rule set, a theory of knowledge organization that would enable us to scale up RBSs without loss of intelligibility or performance, high-grade rule compilers and specialized hardware accelerators, and methods for integrating RBSs easily and seamlessly with conventional DP systems.

The near-term research objectives in RBS technology include integration of RBSs with conventional software systems; modularization and reuse of RBS components; and sharability of RBS knowledge bases among several related applications. Beyond these near-term objectives, long-range goals in RBS research focus on improved hardware for RBS storage and execution tasks, improved standard software and algorithms for common RBS functions, improved architectures that use metaknowledge efficiently, automatic translation of diverse

**Table 3. A Representative Set of RBS Software Products.**

	Product	Host	Vendor
Commercial development	M.1	IBM PC	Teknowledge
	OPS	VAX	Digital
	S.1	VAX, Symbolics	Teknowledge
Research and experimentation	ART	Symbolics	Inference
	ROSIE	VAX, Xerox	RAND

forms of knowledge into rule form, and optimizing compilers for RBSs that perform global data-flow optimizations and exploit multiprocessor opportunities.

### General References

The readings suggested here span a range from introductory to state of the art. Brownston et al. review extensively techniques for using OPS5, a relatively simple but mature rule-based programming system. The Duda and Gaschnig article is a readable, simple introduction. The Erman et al. paper describes the motivations behind S.1 and the techniques used to distinguish analytical rules from imperative prescriptions. *Building Expert Systems* surveys a variety of issues in representing and implementing know-how, including but not limited to RBSs. Shortliffe's book describes one of the seminal projects in RBS technology. Waterman and Hayes-Roth's book surveys the field of RBSs and the closely related but more general pattern-directed inference systems.

- F. Hayes-Roth, "Rule-Based Systems," *CACM* **28**, 921-932 (Sept 1985).
- L. Brownston, R. Farrell, E. Kant, and N. Martin, *Programming Expert Systems in OPS5*, Addison-Wesley, Reading, MA, 1985.
- R. O. Duda and J. G. Gaschnig, "Knowledge-based expert systems coming of age," *Byte*, **6**(9), 238-278 (1981).
- L. E. Erman, A. C. Scott, and P. E. London, Separating and Integrating Control in a Rule-Based Tool, *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, Denver, CO, 1984.
- F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, *Building Expert Systems*, Addison-Wesley, Reading, MA, 1983.
- E. H. Shortliffe, *Computer Based Medical Consultations: MYCIN*, Elsevier, New York, 1976.
- D. A. Waterman, and F. Hayes-Roth, *Pattern-Directed Inference Systems*, Academic Press, New York, 1978.

F. HAYES-ROTH  
Teknowledge

## S

### SAINT

SAINT is a symbolic automatic integrator written by Slagle in 1961 at MIT using the technique of heuristic search in an AND/OR graph (qv) of goals [see J. Slagle, A Heuristic Program that Solves Symbolic Integration Problems in Freshman Calculus: Symbolic Automatic Integrator (SAINT) Report No. 5G-0001, Lincoln Laboratory, MIT, Cambridge, MA, 1961, and J. Slagle, "A heuristic program that solves symbolic integration problems in freshman calculus," *JACM* **10**, 507-520 (1963)].

A. HANYONG YUHAN  
SUNY at Buffalo

### SAM

A script-based (see Scripts) story-understanding system (see Story analysis), SAM was written in 1978 by Cullingford, R. at Yale University (see R. Cullingford, SAM, in R. C. Schank and C. K. Riesbeck (ed.), *Inside Computer Understanding: Five Programs Plus Miniatures*, Erlbaum, Hillsdale, NJ, pp. 75-119, 1981).

M. TAIE  
SUNY at Buffalo

## SCALE-SPACE METHODS

### Qualitative Description and the Problem of Scale

The waveform depicted in Figure 1 typifies a broad class of intricate, ill-behaved physical signals—this one happens to be an intensity profile from a natural image. In what primitive

terms does one talk about, reason about, learn about, and interpret signals like this one? It is universally agreed that the raw numerical signal values are the wrong terms to use for tasks of any sophistication. Plainly, one's initial description ought to be as compact as possible, and its elements should correspond closely to meaningful objects or events in the signal-forming process. Unstructured numerical descriptions are inadequate on both counts.

In the case of one-dimensional signals, local extrema in the signal and its derivatives—and intervals bounded by extrema—have frequently proven to be useful descriptive primitives, although local and closely tied to the signal data, these events often have direct semantic interpretations, e.g., as edges in images. A description that characterizes a signal by its extrema and those of its first few derivatives is a qualitative description of exactly the kind one was taught to use in elementary calculus to "sketch" a function (Fig. 2). Analogous topographic features for two-dimensional functions have been catalogued in Ref. 1. This entry treats the one-dimensional case only; an extension of the methods presented here to two dimensions is an active research topic.

A great deal of effort has been expended to obtain this kind of primitive qualitative description, both in one dimension and two (for overviews of this literature, see Refs. 2-4), and the problem has proved extremely difficult. The problem of scale



Figure 1. Typically complicated natural waveform (an intensity profile drawn from an image).



**Figure 2.** Function characterized by its extrema and those of its first derivative.

has emerged consistently as a fundamental source of difficulty. Any nontrivial local measurement—including a derivative operator—must depend on the value of the signal at two or more points situated on some neighborhood around the nominal point of measurement. The measurement thus depends not only on the signal itself but on the spatial extent of this neighborhood, i.e., on a parameter of scale. Since the scale parameter influences the measured derivatives, it also influences the qualitative description obtained from the derivatives' extrema. In short, one cannot get any description without specifying the scale of measurement, and different scales yield different descriptions.

How does one decide which scale, and therefore which description, is correct? The answer, it appears, is that no single scale is categorically correct; the physical processes that generate signals such as images act at a variety of scales, and none is intrinsically more interesting or important than another. The problem is less to distinguish meaningful events from meaningless noise than to distinguish one process from another, to organize what is seen in a manner that as nearly as possible reflects the physical and causal structure of the world. This and similar lines of thinking have sparked considerable interest in the scale problem and in multiscale descriptions (e.g., see Refs. 3, 5–8).

**Related Work.** A radical view of the scale problem was put forward by Mandelbrot (9), who argues that many physical processes and structures are best modeled by a class of nondifferentiable functions called fractals. Mandelbrot takes the stance that such familiar notions such as length, surface area, slope, surface orientation, etc., ought to be abandoned entirely in favor of global measures of the processes' behavior over scale (e.g., rate of change of measured arc length). Fractal models often capture striking regularities in seemingly chaotic processes, as evidenced by their utility in computer graphics. Pentland (10) has recently argued convincingly for their usefulness in perception as well. However, fractal models cannot supplant the more usual representation of shape: to the mountain climber, knowing what kind of mountain he's on is of little help. He needs to know where next to place his hands and feet.

Marr (11) argued that physical processes act at their own intrinsic scales—e.g., a regular patchwork of wheatfields, a stalk of wheat, and the grains on a stalk of wheat—and that each should be described separately. He described the image by the zero crossings in its convolution with the Laplacian of a Gaussian, at several fixed scales, apparently in the expectation that these channels would cleanly separate physically distinct processes. Unfortunately, in our experience, they generally fail to do so. Indeed, given that the channels' scales are fixed globally and without regard to the structure of the data, it would be astonishing if they succeeded. One is left with a collection of apparently unrelated descriptions, none of which is quite right. The prospect of deciding which description to

use when—or worse still, of picking and choosing among the various channels to build meaningful descriptions—is unappealing. Additionally, Marr proposed the “coincidence assumption” stating that only features that spatially coincide at all scales are physically significant, although no justification was offered for this idea.

Hoffman (12) recently undertook to replace the fixed-scale sequence of descriptions by descriptions at one or more “natural” scales: Measuring the tangent to a curve as a function of scale, he selects as natural scales those points at which the tangent varies least with respect to scale. This approach may offer a significant improvement over fixed channels.

**Scale-Space Approach.** Scale-space filtering addresses two distinct problems (drawing a distinction that Marr and others failed to draw clearly.) First, there are different extrema at different scales, any of which might prove meaningful. How then are the extrema fully characterized over a broad range of scales? Must one find and describe them independently in each channel, as did Marr, or is it possible to construct a more organized and unified description?

Second, how shall these primitives, whatever they turn out to be, be grouped or organized to best reflect the organization of the generating process? Scale is just one among many bases for grouping—along with symmetries, repetitions, etc. Where processes really are cleanly separated by scale, one might hope to use a scale to group their constituent events; however, scale cannot do the whole job. (For an extended discussion of perceptual organization, see Refs. 13 and 14.)

**Scale-Space Description.** One solution to the first problem—that of building a unified description encompassing a broad range of scales—begins with the observation that the descriptions obtained at nearby scales appear to have a great deal in common: One often sees extrema in nearly the same locations, although one is likely to see a number of extrema in the finer channel with no counterparts in the coarser one. (Presumably the same observation motivated Marr's “coincidence assumption.”) Is there any well-defined sense in which extrema observed at different scales can be said to correspond, to manifest the same event seen through different filters, rather than two unrelated events? This question is answered by treating scale as a continuous parameter, considering the effect of a very small scale change in scale. (A scale parameter is taken as the standard deviation,  $\sigma$ , of a Gaussian convolved with the signal.) As  $\sigma$  is decreased continuously, beginning at a coarse scale, one observes two distinct effects on the extrema: existing extrema move continuously along the signal axis and new extrema occasionally appear at singular points. These effects are best visualized in terms of the surface swept out on the  $(x, \sigma)$  plane by varying  $\sigma$ . The  $(x, \sigma)$  plane is called scale space, and the surface, the scale-space image of the signal. The extrema, viewed on the scale-space image, form contours. The tops of these contours are the singularities, the points above which a continuously moving extremum vanishes (look ahead to Fig. 6.)

On the assumption that each of these contours in general reflects a single physical event, the contour, rather than its constituent points, is interpreted as the unit of description. The scale of the event is the scale at which the contour vanishes, and its location in the signal domain is its location at the finest observable scale. The raw qualitative description thus produced consists of a single sequence of extrema, each observ-



able on a definite range of scales, and each referred to a definite signal-domain location. This unified representation exhaustively describes the qualitative structure of the signal over all observed scales. Each element of the description has a scale associated with it (the vanishing scale), but the description itself is not partitioned into arbitrary channels or levels. The description makes the scale of each event available as a basis for organization but does not prematurely impose a rigid organization.

**Organization and Scale.** Consider the role of scale as a means of organizing the raw description. A scale-structured representation, called the interval tree, is introduced. Building on the raw scale-space description, and in particular on the singular points at which new extrema appear, the interval tree captures the coarse-to-fine unfolding of finer and finer detail. The tree is used to generate a set of descriptions, varying the scale in local, discrete steps that reflect the qualitative structure of the signal. Although highly constrained, this family of descriptions appears to capture perceptually salient organizations.

Additionally, a stability criterion is applied to produce "top-level sketches" of signals as a starting point for matching and interpretation.

### Scale-Space Image: Definition

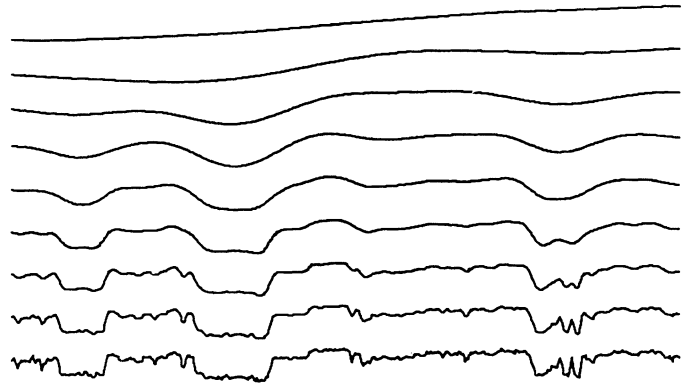
A parameter of scale is introduced to define what is meant by "the slope at point  $x$  and scale  $\sigma$ ." The signal is convolved (enfolded) with a Gaussian, and the Gaussian's standard deviation is taken to be the scale parameter. Although scale-dependent descriptions may be computed in many ways, the Gaussian convolution is attractive for a number of its properties, amounting to "well-behavedness": The Gaussian is symmetric and strictly decreasing about the mean, and therefore the weighting assigned to signal values decreases smoothly with distance. The Gaussian convolution behaves well near the limits of the scale parameter,  $\sigma$ , approaching the unsmoothed signal for small  $\sigma$ , and approaching the signal's mean for large  $\sigma$ . The Gaussian is also readily differentiated and integrated.

Although the Gaussian is not the only convolution kernel that meets these criteria, a more specific motivation for this choice is a property of the Gaussian convolution's zero crossings (and those of its derivatives): As  $\sigma$  decreases, additional zeroes may appear, but existing ones cannot in general disappear; moreover, of convolution kernels satisfying the well-behavedness criteria (roughly those enumerated above,) the Gaussian is the only one guaranteed to satisfy this condition (15). Recently this result has been extended to two-dimensional Gaussian convolutions (16). This is an important property because it means that all the extrema observed at any scale are observable at the finest scale, which, as shown below, greatly simplifies the description.

The Gaussian convolution of a signal  $f(x)$  depends both on  $x$ , the signal's independent variable, and on  $\sigma$ , the Gaussian's standard deviation. The convolution is given by

$$F(x, \sigma) = f(x) * g(x, \sigma) = \int_{-\infty}^{\infty} f(u) \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-u)^2/2\sigma^2} du \quad (1)$$

where  $*$  denotes convolution with respect to  $x$ . Here  $F$  defines a surface on the  $(x, \sigma)$  plane, the surface swept out as the Gaussian's standard deviation, is smoothly varied. The  $(x, \sigma)$ -plane



**Figure 3.** Sequence of Gaussian smoothings of a waveform, with  $\sigma$  decreasing from top to bottom. Each graph is a constant- $\sigma$  profile from the scale-space image.

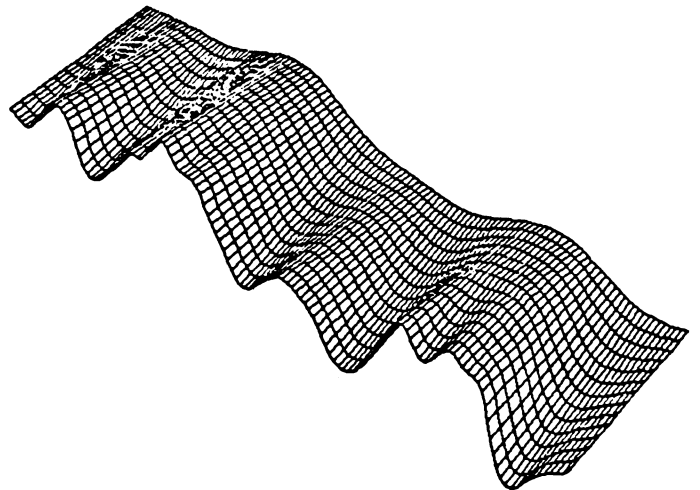
scale space and the surface  $F$  defined in Eq. 1 are called the scale-space image of  $f$ . It is actually convenient to treat  $\log \sigma$  as the scale parameter, as uniform expansion or contraction of the signal in the  $x$  direction will cause a translation of the scale-space image along the  $\log \sigma$  axis. All illustrations portray  $\sigma$  on a log scale. Figure 3 graphs a sequence of Gaussian smoothings at increasing  $\sigma$ , which are constant- $\sigma$  profiles from the scale-space image. Figure 4 portrays the scale-space image as a surface in perspective.

### Scale-Space Image: Qualitative Structure

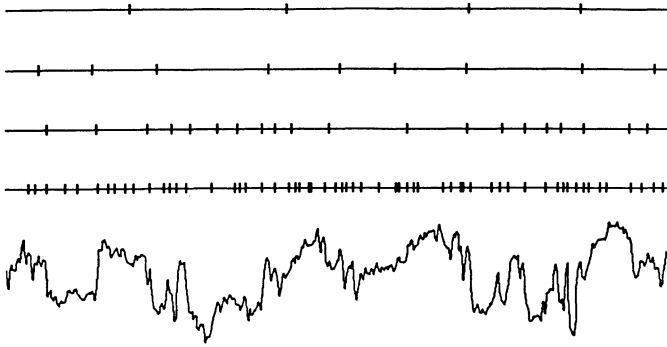
An extremum in the  $n$ th derivative is a zero crossing in the  $(n + 1)$ th. Although, conceptually, one is interested in extrema, the zero crossings are more convenient to work with. The  $n$ th-order zero crossings in  $F$  are the points that satisfy

$$\frac{\partial^n F}{\partial x^n} = f * \frac{\partial^n g}{\partial x^n} = 0 \quad \frac{\partial^{(n+1)} F}{\partial x^{(n+1)}} \neq 0 \quad (2)$$

where the derivatives of the Gaussian are readily obtained. Only the partials with respect to  $x$ , not  $\sigma$ , correspond to zero crossings in the smoothed signal at some scale. These points are extrema in the  $(n - 1)$ th derivative. Thus, evaluating the



**Figure 4.** Gaussian smoothing sequence of Figure 3 portrayed as a surface in perspective.



**Figure 5.** Typical waveform, with the locations of its second-order zero crossings taken at several scales.

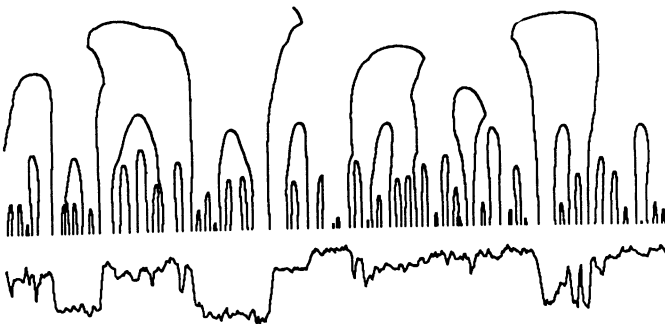
partials at any fixed  $\sigma$ , the zero crossings in  $\partial F/\partial x$  are local minima and maxima in the smoothed signal at that  $\sigma$ , those in  $\partial^2 F/\partial x^2$  are extrema of slope (inflections), and those in  $\partial^3 F/\partial x^3$  are extrema of (unnormalized) curvature.

Sampling  $F$  along several lines of constant  $\sigma$  yields a filter bank of the sort used by Marr and others. If the qualitative descriptions were computed separately for each slice, one would face the basic scale problem discussed earlier, i.e., one would be confronted with a different description at each scale, having no clear basis for relating one to another, or deciding which to use when. One wishes to avoid this chaotic state of affairs.

The scale-space image, because it treats  $\sigma$  as a continuous variable, offers the means to do so. Figure 5 shows a typical waveform with the zero crossings in  $\partial^2 F/\partial x^2$  (inflection points) taken at several values of  $\sigma$ . Some of the inflections appear to correspond over scale, although they are not exactly aligned. Others seem to have no correspondence, or ambiguous correspondence. Is there any meaningful sense in which certain extrema correspond over scale? Figure 6 shows what happens when the gaps are filled in: The fixed-scale zeroes in fact lie on zero-crossing contours through scale-space. Only some of the isolated zeroes observed in Figure 5 do correspond, in the sense that they lie on the same contour.

Observe that these contours form arches, closed above but open below. At the apex of each arch,  $F$  satisfies

$$\frac{\partial^n F}{\partial x^n} = 0 \quad \frac{\partial^{n+1} F}{\partial x^{n+1}} = 0 \quad \frac{\partial^{n+1} F}{\partial x^n \partial \sigma} \neq 0 \quad (3)$$



**Figure 6.** Contours of  $F_{xx} = 0$  in the scale-space image. The  $x$  axis is horizontal; the coarsest scale is on top. To simulate the effect of a horizontal scale change on the qualitative description, hold a straight-edge (or better still, a slit) horizontally. The intersections of the edge with the zero contours are the zero crossings at some single value of  $\sigma$ . Moving the edge up or down increases or decreases  $\sigma$ .

which is not a zero-crossing point by the definition in Eq. 2. Thus a "contour" really means a single arm of an arch, with the apex deleted. Each contour may be viewed as a zero crossing that moves continuously on the  $x$  axis as  $\sigma$  is varied, with instantaneous "velocity"

$$-\frac{\partial^{n+1} F / \partial x^n \partial \sigma}{\partial^{n+1} F / \partial x^{n+1}} \quad (4)$$

The denominator in Eq. 4 is the slope at which  $\partial^n F / \partial x^n$  crosses zero (the "strength" of the zero crossing), and the numerator responds to features entering or leaving the mask's effective receptive field. Thus, the location of strong zero crossings tends to be more stable over scale than that of weak ones, but any zero crossing may be "pulled" by sufficiently large surrounding features. Also, as the strength of a zero crossing approaches zero, its velocity may become arbitrarily large.

The two arms of each arch form a complementary pair, crossing zero with opposite sense. As one sweeps across the apex of an arch, with  $\sigma$  increasing, the pairs approach each other with increasing velocity and then collide and are annihilated. The proof of Ref. 15, mentioned above, is assurance that the complementary singularity—a pair of zeroes vanishing as one moves to a finer scale—can never occur.

#### "Raw" Scale-Space Description

Thus, a decrement in  $\sigma$  has two effects on the zero crossings: the continuous motion of existing zero-crossings, according to Eq. 4, and the appearance of new ones, in complementary pairs, at the singular points described in Eq. 3. A zero crossing observed at any scale may always be tracked continuously across all finer scales.

What do these properties imply for qualitative description? Suppose the  $f$  at a particular scale is qualitatively described by the sequence of zero crossings at that scale, and for each zero-crossing the sense with which it crosses zero, as well as the order of the extremum it represents, is noted (if more than one order is being used). Each of these fixed-scale zeroes lies on a zero-crossing contour in scale space. The contours, once they appear, cannot disappear at finer scales, in general change their sense, or cross each other. Therefore, the initial sequence must always be embedded, in its entirety, in the sequence obtained at any finer scale. The only changes in the qualitative description as  $\sigma$  decreases occur at the "apex" singularities, where a new pair of zero crossings must be spliced into the sequence.

Therefore, one may capture the qualitative description at all scales just by recording the sequence of zero-crossing contours intersecting the finest observable scale and, for each, the value of  $\sigma$  at which the contour vanishes. The qualitative description at any scale is then obtained by deleting from the full sequence the zero crossings that vanish at a finer scale. Rather than a collection of independent single-scale descriptions, this affords a unified description, each of whose elements exists over a definite range of scales. Complete quantitative information may be attached to the qualitative description by recording the contours' trajectories through scale space and relevant properties of  $F$  along the trajectory.

**Identity, Scale, and Localization Assumptions.** The scale-space description, defined above, captures the qualitative structure of the signal at all scales. But what physical interpretation shall be placed on the events comprising the descrip-

tion? What does a scale-space contour mean, as well as the geometry of its trajectory, and so forth?

One can first assume that each contour reflects a single physical event. When one eventually says, e.g., that "this inflection exists because. . .," one must complete the sentence in the same way for every point on the contour. This just amounts to an assumption that the organization of the scale-space image is physically meaningful, that when  $\sigma$  is changed a small amount, one is almost always seeing the same broad-band events at a different scale rather than a whole new set of narrow-band events. This "identity" assumption allows one to treat each contour as an indivisible unit for the purpose of interpretation.

The scale of each event, for the purpose of grouping and organization, is assumed to be given by the "apex" singularity, the scale above which the contour vanishes. Thus, a "large-scale event" is one that is observable at a large  $\sigma$  and all smaller ones.

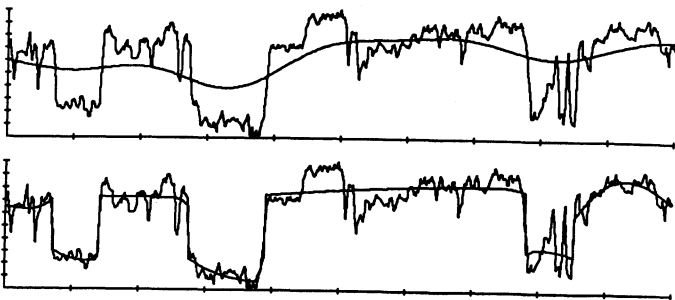
Finally, the true location of a zero-crossing contour on the  $x$  axis is assumed to be its location at the finest observable scale. This assumption is motivated by the observation that linear smoothing has two effects: qualitative simplification (the removal of fine-scale features) and spatial distortion (dislocation, the broadening and flattening of the features that survive). The latter effect is undesirable, because a big event is not necessarily a fuzzy one. Because each event in the scale-space description extends over a range of scales, there is no difficulty in assigning a fine-scale location to a coarse-scale event.

Collectively, these assumptions characterize each scale-space contour as denoting a single physical event whose scale is the contour's vanishing scale and whose location is the contour's fine-scale location.

Figure 7 illustrates the effectiveness of this interpretation: One selects the events whose scale exceeds a threshold and then draws a "sketch" of the signal by fitting parabolic arcs between the distinguished points. This description is qualitatively isomorphic to the Gaussian-smoothed signal at the same scale, but it much better preserves the locations of the distinguished points.

### Organization and Scale

This entry began by distinguishing the problem of characterizing primitive events over scale from the problem of grouping



**Figure 7.** Below is shown a waveform with a superimposed approximation. The approximation was produced by identifying inflections at a coarse scale and resolving their fine-scale locations. Parabolas were then fit independently between each pair of localized inflections. Above is shown the corresponding (qualitatively isomorphic) Gaussian smoothing.

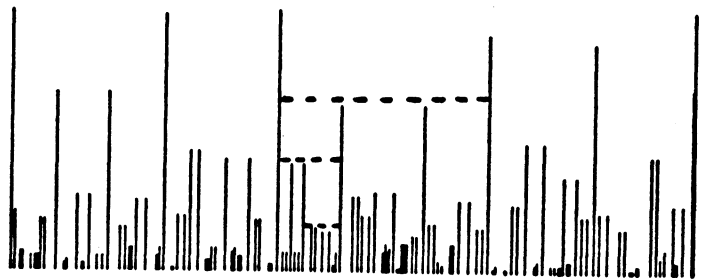
or organizing them in a meaningful way. The raw scale-space description, by design, is noncommittal with respect to organization: It concisely lists the primitive events observable at all scales, each with its scale and location, as well as other qualitative and quantitative information. Given the raw description, one is free to use the events' scale, as well as the rest, in whatever manner one likes. In this section a representation is introduced, derived from the raw scale-space description, that is organized by scale. This representation, called the interval tree, describes the division of intervals bounded by extrema into finer and finer subintervals, as  $\sigma$  decreases. The scale channels proposed by Marr and others produce poor organizations because events are structured by a series of arbitrary global-scale thresholds. In contrast, the interval tree permits the scale of description to be controlled in local discrete steps determined by the qualitative structure of the signal.

**Interval Tree.** The scale-space extrema whose scales exceed a given threshold  $\sigma_T$  partition the  $x$  axis into intervals. As  $\sigma_T$  is decreased, starting from a coarse scale, new events appear in pairs (each associated with an "apex" singularity in the scale-space image), dividing the enclosing interval into a triple of subintervals (see Fig. 8.) As  $\sigma_T$  is decreased further, these new intervals in turn subdivide, down to the finest observable scale.

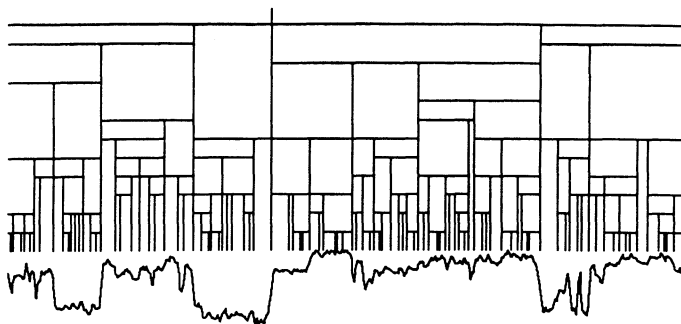
Each of these intervals defines a rectangle in scale space bounded above by the scale at which the interval emerges out of an enclosing one, bounded below by the scale at which it splits into subintervals, and bounded on either side by the  $x$  locations of the events that define it. Thus, each interval has a definite width and location on the signal axis and exists over a definite range of scales. Collectively, the rectangles tessellate the  $(x, \sigma)$  plane.

The intervals also correspond to nodes in a ternary-branching tree: An interval's parent is the larger interval from which it emerged, and its offspring are the subintervals into which it divides. This interval tree is illustrated in Figure 9. (For convenience, a dummy interval including the whole signal is taken as the root of the tree.)

**Using the Interval Tree.** The coverings of the  $x$  axis by the interval tree (i.e., all the sets of nodes for which every point on the  $x$  axis is covered by exactly one node) each segment the signal into a set of intervals. From any starting point, one may generate a new segmentation either by splitting an interval into its offspring, or merging some intervals into their common parent. The space of descriptions may be thus explored,



**Figure 8.** Vertical lines represent a sequence of distinguished points, with the height of each line representing the scale at which the event vanishes. The events whose scale exceeds a threshold,  $\sigma_T$ , partition the  $x$  axis into intervals. As we decrease  $\sigma_T$ , new events appear in pairs, dividing the enclosing interval into a triple of subintervals.

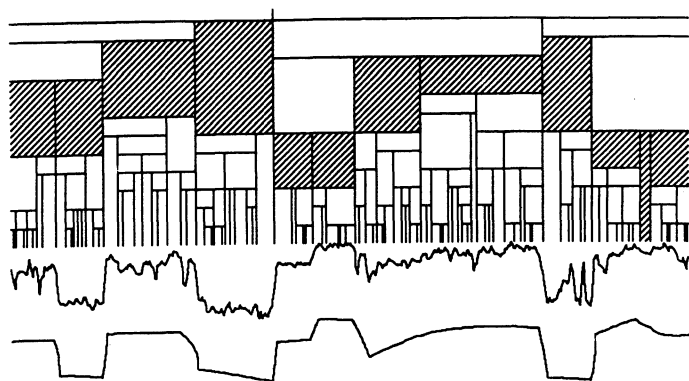


**Figure 9.** A signal with its interval tree, represented as a rectangular tessellation of scale space. Each rectangle is a node, indicating an interval on the signal and the scale range over which the signal interval exists.

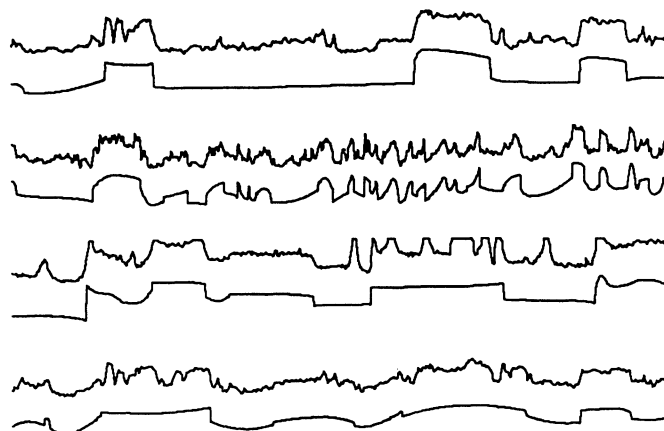
where the scale of description is changed in local, discrete steps (see Fig. 10.) As a result, one has collapsed the  $(x, \sigma)$  plane into a discrete set, taking advantage of the singular points at which extrema appear to do so.

The set of segmentations generated by the interval tree is a small fraction of the segmentations one could generate by collecting arbitrary subsets of all the scale-space extrema. In particular, a segmentation is excluded if any of its intervals contains an extremum of larger scale than either of the two bounding ones. This "interval constraint" is in fact a useful and conservative one because it discards the myriad of meaningless groupings—constructed, e.g., by joining pairs of widely separated fine-scale extrema, skipping an arbitrary number of large-scale ones along the way—and preserves "smooth" intervals that are broken only by extrema of much finer scales than the bounding ones. It has been found, by extensive but informal test, that people are nearly always able to duplicate the segmentations they find perceptually salient within this constraint by moving interactively through the tree. Figure 11 shows a few signals with "sketches" that have been obtained in this way.

**Stability Criterion.** Although the interval tree generates descriptions in an organized way, it would be useful to establish a preference ordering on those descriptions to obtain "top-level" sketches as starting points for matching and interpretation. The interval tree initially requires that all extrema within each interval be of finer scale than the bounding ones. A natural extension to this useful constraint would be to favor



**Figure 10.** Approximation derived from a particular covering of the signal by the interval tree. The "active" nodes (i.e., those that determined the segmentation) are shaded.



**Figure 11.** Descriptions obtained by interactively traversing the interval tree to match perceptually salient segmentations.

intervals for which the difference between the scale of the bounding extrema and that of the largest extremum between them is large. In the interval tree this difference is just a node's extent in the scale domain—its persistence or stability. A very stable node is one bounded by very large-scale extrema but containing only very small ones—intuitively, a "smooth" interval.

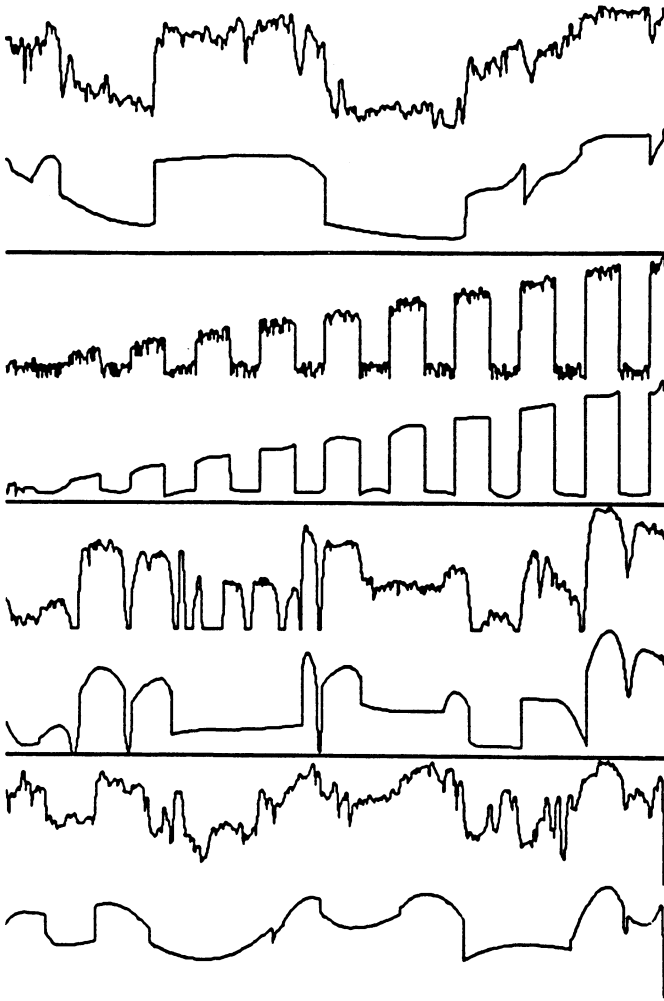
Maximum stability descriptions are acquired by starting from the root of the tree and moving down recursively whenever a node's stability is less than the mean stability of its offspring. These are local maxima, which may be layered at successively finer scales. (Several variants on this procedure achieved similar results.) Figure 12 shows several signals with their top-level maximum stability descriptions. If these are compared with one's own percepts, it can be observed that they closely resemble the interactively obtained descriptions of Figure 11.

## Conclusions

The entry set out with two objectives: to obtain a unified description of the extrema in a signal over a wide range of scales and to use the scale of extrema to constrain and guide their organization and grouping. Both objectives were attained.

**Description.** The extrema observed at a particular scale move continuously as the scale is varied, vanishing in pairs at singular points as the scale becomes coarser. These continuously moving extrema form contours on the scale-space image, the surface swept out by smoothly varying the scale. These contours are presumed to be the units of description on the assumption that each in general denotes a single physical event. The qualitative structure over all observed scales is captured in a single sequence of extrema, each extending over a range of scales and each characterized by its fine-scale location and the scale above which it vanishes.

**Organization.** The least unit of organization is an interval bounded by a pair of extrema. A criterion for meaningful groupings is the difference in scale between the bounding extrema and the ones within the interval, where the bounding extrema are much larger, the interval is a strong candidate for treatment as an "undistinguished interval" (i.e., monotonic in the derivatives in which extrema were found). This grouping

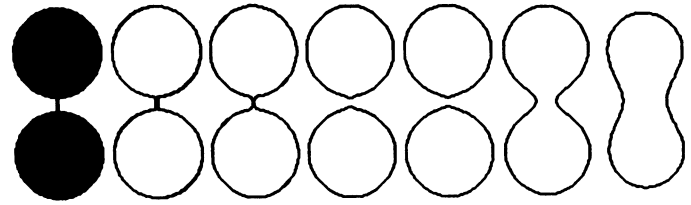


**Figure 12.** Several signals, with their maximum-stability descriptions. These are top-level descriptions, generated automatically and without thresholds. You should compare the descriptions to your own first-glance top-level percepts.

criterion is captured in the interval tree, a ternary-branching structure, simply derived from the bare description, that describes the coarse-to-fine subdivision of intervals. The structure of the tree itself excludes intervals containing larger-scale extrema than the bounding ones. Within this constraint, the persistence or stability over scale of an interval in the tree measures the scale difference. Using this measure, maximum stability groupings were constructed to provide top-level sketches of the signal.

The bare scale-space description and the organization provided by the interval tree constitute a sound basis for further organization by discovering regularities such as symmetry, repetition, and intersignal correspondence; for the construction of larger morphological structures expressed as sequences of extrema; and for graphic communication between man and machine. All of these topics are currently being investigated, and the basic methods are being applied to signal-understanding problems in geology, vision, and speech. Carlotto (17) has used scale-space descriptions and interval trees to describe and parse histograms. A related scale hierarchy for plane curve description was developed by Asada and Brady (18).

Extension of scale-space methods to two dimensions is a topic of considerable interest. In extending the results of (15)



**Figure 13.** The splitting and merging of zero-crossing contours over scale is illustrated by the "dumbbell" shape at left. Black and white represent 1 and -1, respectively. The sequence of contour maps show zeroes in the convolution of the shape with a Gaussian. The standard deviation of the Gaussian doubles with each map, from left to right. At both the finest and coarsest scales, a single contour divides the plane into two regions. On an intermediate range the contour splits into two, dividing the plane into three regions.

to two dimensions, Yuille and Poggio (16) showed that zero-crossing contours (and other level curves) in a Gaussian-convolved image cannot vanish with decreasing scale. Yuille and Poggio (19) addressed the related question of reconstructing a signal from its zero crossings over scale, and Hummel et al. (20) addressed the related problem of using the scale-space image to reverse the effect of Gaussian blur.

Although in one dimension this monotonicity condition guarantees that an interval tree can be built, the situation in two dimensions is unfortunately more complicated. Zero-crossing contours, although they may not vanish, are free to split and merge at increasingly fine scale. The regions bounded by zero-crossing contours therefore split and merge as well and their behavior over scale may not be described by a simple tree. The resulting difficulties are illustrated in Figure 13, which shows a simple "dumbbell" pattern and the zeroes of its Gaussian convolution at several scales. At both fine and coarse scales a single contour divides the plane into two regions. On an intermediate range the contour splits in two, dividing the plane into three regions.

An alternative approach to the two-dimensional extension is given by Kass and Witkin in Ref. 21. The orientation structure of the image is used to build a coordinate system in which curved edges typically map to straight lines at fixed orientations. Because the shapes and orientations of edges in this new coordinate system are approximately known, one-dimensional scale-space descriptions can be computed along and across the edge, reducing the dimensionality of the scale-space description.

## BIBLIOGRAPHY

1. R. M. Haralick, L. T. Watson, and T. J. Laffey, *The Topographic Primal Sketch Technical Report*, Computer Science Department, Virginia Polytechnic Institute, 1982.
2. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, New York, 1976.
3. D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice Hall, Englewood Cliffs, NJ, 1982.
4. T. Pavlidis, *Structural Pattern Recognition*, Springer-Verlag, NY, 1977.
5. A. Rosenfeld and M. Thurston, "Edge and curve detection for visual scene analysis," *IEEE Trans. Comput.* **C-20**, 562-569 (May 1971).
6. D. Marr and E. C. Hildreth, *Theory of Edge Detection*, MIT Artifi-

- cial Intelligence Memo 518, Cambridge MA, April 1979; *Proc. Roy. Soc. B* **207**, 187–217 (1980).
7. D. Marr and T. Poggio, MIT AI Memo 451, 1977; "A computational theory of human stereo vision," *Proc. Roy. Soc. Lond. B* **204**, 301–328 (1979).
  8. T. H. Hong, M. Shneier, and A. Rosenfeld, Border Extraction Using Linked Edge Pyramids, TR-1080, Computer Vision Laboratory, University Maryland, July 1981.
  9. B. Mandelbrot, *Fractals: Form, Chance, and Dimension*, W. H. Freeman, San Francisco, 1977.
  10. A. P. Pentland, Fractal-Based Description, *Proceedings of the Eighth IJCAI*, Karlsruhe, FRG, 1983, pp. 973–981.
  11. D. Marr, *Vision*, W. H. Freeman, San Francisco, 1982.
  12. D. Hoffman, Representing Shapes For Visual Recognition Ph.D. Thesis, MIT, Cambridge, MA, June 1983.
  13. A. P. Witkin and J. M. Tenenbaum, "On the Role of Structure in Vision," in A. Rosenfeld (ed.), *Human and Machine Vision*, Academic Press, New York, 1983, pp. 481–543, 1983.
  14. A. P. Witkin and J. M. Tenenbaum, What is Perceptual Organization For? *Proceedings of the Eighth IJCAI*, Karlsruhe, FRG, 1983, pp. 1023–1026.
  15. J. Babaud, M. Baudin, A. Witkin, and R. Duda, Uniqueness of the Gaussian Kernel for Scale-Space Filtering, Fairchild Technical Report No. 645, 1983; and *PAMI* **8**(1), 26–33 (1986).
  16. A. L. Yuille and T. Poggio, "Scaling theorems for zero-crossings," *PAMI* **8**(1), 15–25 (1986).
  17. M. J. Carlotto, Histogram Analysis Using a Scale-Space Approach, *Proceedings of the CVPR*, 1985, pp. 334–340.
  18. H. Asada and M. Brady, The Curvature Primal Sketch, MIT AI Memo 758, February 1984.
  19. A. L. Yuille and T. Poggio, Fingerprints Theorems, *Proceedings of the Fourth AAAI*, Austin, TX, 1984, pp. 362–365.
  20. R. A. Hummel, B. Kimia, and S. W. Zucker, Deblurring Gaussian Blur, McGill University, Technical Report 83-13R, 1984.
  21. M. Kass and A. Witkin, Analyzing Oriented Patterns, *Proceedings of the Ninth IJCAI*, Los Angeles, 1985, pp. 944–952.
  22. A. P. Witkin, Scale Space Filtering, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, 1983, pp. 1019–1022.
  23. A. P. Witkin, Scale Space Filtering: A New Approach to Multi-Scale Description, in S. Ullman and W. Richards (eds.), *Image Understanding 1984*, Ablex, Norwood, NJ, 1984, pp. 79–95.

### General References

- A. Blumenthal, L. Davis, and R. Rosenfeld, "Detecting natural 'plateaus' in one-dimensional patterns," *IEEE Trans. Comput.*, 178–179 (February 1977).
- R. Erich and J. Foith, "Representation of random waveforms by relational trees," *IEEE Trans. Comput. C-26*, 725–736 (July 1976).

A. WITKIN  
Fairchild

## SCHOLAR

The SCHOLAR system is a set of programs written by Carbonell (J. R. Carbonell, Mixed-Initiative Man-Computer Instructional Dialogues, Ph.D. Dissertation, MIT, Cambridge, MA, June 1970). SCHOLAR, like older CAI (computer-assisted instruction (qv)) programs, simulates a teacher, but it permits the student to ask questions and in that sense is a mixed-initiative system. SCHOLAR acts as a geography

teacher; however, its network-based knowledge representation (qv) makes it possible to change the teaching subject with only minor changes to the executive program (for an overview, see J. R. Carbonell "AI in CAI: An artificial intelligence approach to computer-assisted instruction," *IEEE Trans. Man-Machine Syst. MMS-11*(4), 190–202 (November 1970)).

J. GELLER  
SUNY at Buffalo

## SCRIPTS

In this entry the notion of script (1) is discussed as a memory structure that organizes knowledge about stereotypical situations, such as dining at a restaurant, going to the movies, or shopping for groceries. The entry focuses on the structure of scripts, script application, problems with scripts, psychological validity of the script notion, and issues in script acquisition. The process of script application is discussed from the perspective of four story-understanding computer programs.

A script (1) is a knowledge structure containing a stereotypic sequence of actions. Scripts encode culturally shared knowledge of stereotyped actions that occur in socially ritualized activities, such as going to stores, restaurants, and museums; riding trains and subways; attending plays and banquets; and playing games or driving cars. Scripts are intended to capture situations in which the behavior is so stylized that the need for complex goal and plan analysis rarely arises. People probably acquire scripts through repeated exposure to stereotypic situations. Scripts are knowledge constructs that tell people what can happen in a situation, what events follow, and what roles various people are expected to play in a given social setting. In addition to guiding behavior, scripts are of value in directing cognitive processes, e.g., while reading narratives that involve social activities.

An example of a prototypical script is \$RESTAURANT, which captures the activities involved in eating in a restaurant. The \$ distinguishes the script from the word "restaurant," which refers to the physical setting in which the scriptal activities occur. It is important to realize that the notion of a script is independent of any natural language used to describe it. For example, the event of being seated in a restaurant is described using different words or phrases in Spanish, but the events themselves are the same. Of course, the content of a script will vary by subculture since different social groups have distinct forms of ritualized behavior. For example, the events in \$WEDDING will differ in African cultures. However, all people probably access and apply knowledge constructs that serve the function of scripts. From this point of view, the notion of a script is a cultural universal.

In \$RESTAURANT, e.g., the ordinary course of affairs is that the patron enters, is seated, and orders a meal. The meal is then prepared and served, and the patron eats it. Finally, the patron pays the bill and leaves. Each of these activities is described by a stereotyped sequence of events, which prescribes the order in which things happen and the people and objects participating in the action. Entering the restaurant, looking for a table, walking over to one, and sitting down comprise one such event chain. Each event has resulting states that in turn become the enabling conditions for further events to occur. For example, one must be physically inside the restaurant before one can look for a table. Seeing an empty



table enables walking over to it. As a result of walking to a table, one can sit down at it.

Scripts help both humans and computers process text involving stereotypical situations. Consider the following script-based story:

Mary went to a restaurant. She ordered lobster.  
She left a large tip.

\$RESTAURANT provides the context necessary for inferring events not explicitly mentioned in the story (e.g., eating lobster and paying the check) and for recognizing the causal relationships among events explicitly mentioned in the story (e.g., the relationship between ordering lobster and eating it).

In this entry the concept of script and its application in natural-language processing is discussed with emphasis on the structure of a script; use of, and problems with, scripts; psychological validity of scripts; and the script-acquisition process. In addition, the uses of the notion of script by the computer systems SAM, FRUMP, BORIS, and OpEd are described.

### Structure of a Script

What does a script look like internally (2,3)? Consider the subway script (e.g., in New York). A patron enters the station and goes to a turnstile. Next the patron puts a token in, passes through, and goes to the appropriate platform. Eventually, the train comes. The patron enters and finds a seat. After a number of stops the destination is reached, and the patron leaves the train and exits from the station.

This stereotyped sequence of events in the "backbone" of the subway script is understood and used by millions of commuters in New York and, with minor variations, in other cities as well. In each case, there is an organization (the subway company or authority) providing a certain kind of transportation to a member of the public in return for money. In \$SUBWAY there is a cast of characters ("roles"), the objects they use while going about their business ("props"), and the places ("setting") where the script's activities happen. The roles, props, and setting of a script taken together make up the script variables, which are matched up against real-world (or fictitious) people, places, and objects. For instance, the roles of the subway script are:

&PATGRP	A group of subway riders
&CASHIER	The cashier
&CONDUCTOR	The conductor
&DRIVER	The person controlling the train
&SUBORG	The subway organization

For example, the patron role (&PATGRP) in \$SUBWAY must be filled by the class PERSON or GROUP since both "John Smith" and "Mr. and Mrs. Smith" are accepted as role fillers. The subway company providing the service, e.g., "the BMT," must belong to the class ORGANIZATION.

The settings of the script are the places where the script's events happen. Settings belong to the class LOCALE. In \$SUBWAY the three most important settings are the originating station, the inside of the car the patron selects, and the destination station.

The props of a script are associated either with the script's roles or its settings. Role-based props include such small objects as tokens and coins. Setting-based props function as "furniture" in a script. For example, the cashier's booth and the

turnstile are props in the subway concourse; seats and bubble gum machines are props on a platform. A special prop in \$SUBWAY is the train itself. This is an example of "structured" physical object whose parts, the cars, are important locations for script activity in their own right. The props of \$SUBWAY include:

&TOKEN	A token
&FARE	Money paid for a token
&TURNSTILE	A turnstile
&PLATSEAT	A seat on the platform
&SUBWAY	The train itself
&SUBWAYCAR	One of the cars
&CARSEAT	A seat on the car
&STRAP	A strap for the patron to grasp
&EXITGATE	The gate leading from the platform at the destination station

The most important components of \$SUBWAY are its events, involving the roles, props, and settings. Scriptal events are represented in terms of conceptual-dependency (qv) (CD) structures (4,5). For example, the patron's giving money to the cashier at the cashier's cage is represented as:

(ATRANS	
ACTOR	&PATGRP
OBJECT	&FARE
FROM	&PATGRP
TO	&CASHIER)

where ATRANS represents, in CD notation, an abstract transfer of possession or control.

As with scripts in general, the representation of events in CD are language-free. The CD representation of an event provides a canonical form for mapping many surface strings or inputs that are conceptually equivalent. For instance, the same CD form would be used no matter if the sentence were "John gave a dollar to the cashier," "The cashier got a dollar from John," or "a dollar was received from John by the cashier." The use of CD representation thus cuts down tremendously on the size of the script since only the conceptual content (and not the surface form) of the sentences need be considered. It also reduces the amount of processing since needed inferences can be tied together via conceptual events rather than having to be duplicated for each distinct surface string with an equivalent meaning. For instance, if  $x$  ATRANS object  $o$  from  $x$  to  $y$ , then one can conclude that  $x$  no longer controls  $o$  and  $y$  now does. By representing and factoring out the transfer of control aspect of "give," "get," "buy," "loan," "steal," etc., in terms of an ATRANS predication, one acquires access immediately to those inferences associated with ATRANS. For instance, although "steal" maps to more than ATRANS alone (e.g., stealing implies that it is against the owner's will), that aspect of its representation in terms of ATRANS allows natural-language analyzers to conclude automatically from, say, "John stole the car from Bill" that Bill has lost physical control of the car and John now has control of it.

Script events contain both "constant" parts (e.g., ACTOR and ATRANS in the example given above) and "variable" parts, i.e., they are patterns designed to match an arbitrary range of real-world events (see Pattern matching). For example, any member of the public can ride on the subway, so the corresponding slot in the script's events cannot be fixed but must accept any person or group that comes along in the story. In the "paying the cashier in the subway" activity, one needs a

way to specify the things that are always true. For example, this event has a person handing over an amount of money to another person who is an agent of the subway organization. One also has to provide for things that can vary in small details. The fare may be expressed as "a dollar" or "four quarters." "John" may pay the cashier or "John and Mary" may pay.

The basic idea in defining a pattern is to include the minimum amount of information needed to uniquely identify the event. Consider, for example, the pattern for:

"Patron enters the station"

```
(PTRANS
  ACTOR  &PATGRP
  OBJECT &PATGRP
  TO     (INSIDE
          PART &STATION))
```

where PTRANS is a CD predicate that organizes inferences involving transfer of physical location. This pattern can match conceptualizations that correspond to inputs such as:

1. John and Mary went into a subway station.
2. John walked into a subway station.
3. John strolled out of a restaurant up the street into a subway station.
4. John went into the BMT (Brooklyn-Manhattan Transit).

Example 1 would instantiate the pattern because John and Mary form a group. In example 2 the pattern would consider the fact that John "walked" to the subway as insignificant; it would create the same conceptualization if John had "sauntered," "rambled," "ran," or even "came in on rollerskates." In 3 where John came from is of no interest to \$SUBWAY. (It would constitute a signal that the activated \$RESTAURANT be closed before \$SUBWAY is opened.) Finally, 4 would instantiate the pattern because BMT is a subway organization.

**Script Invocation with Script Headers.** An important class of patterns are called script headers, which act to "invoke" or "instantiate" a script. The basic rule in defining a script header is that a complete event is needed to bring the script into play. For example, \$RESTAURANT should not be invoked just because "a restaurant" is mentioned. This is not to say that script information should be completely suppressed because it may be used in later stages of understanding. For example, in "I met a truck driver in a restaurant," remembering that in \$RESTAURANT the person had a &DRIVER role in \$TRUCKING may be crucial to understanding what he might say or do later.

Conceptualizations can be produced not only by surface clauses but also by certain kinds of prepositional phrases. Such phrases can act as complete thoughts by modifying the time or place setting of the main event. Consider the following sentence:

Mary was killed in an auto accident.

The above phrase can be paraphrased roughly as "When an accident occurred, Mary was killed." The top-level event of Mary's being killed is placed into some temporal relation to the "accident." Thus, just the prepositional phrase "in an auto

accident" is sufficient to create a conceptualization that can invoke the entire \$VEHICLE-ACCIDENT script.

Script headers come in four varieties and are ranked on the basis of how strongly they predict that the associated context will be instantiated. The first type is called a precondition header (PH) because it triggers a script on the basis of a main script precondition being mentioned in the text. As an example, the sentence "John was hungry" is a PH for \$RESTAURANT because it is enabling the condition for the main conceptualization (INGEST food) of the script. A story understander having access to both scripts and plans would make the (relatively weak) prediction that \$RESTAURANT might come up because this is known to be a common means (i.e., a plan) of getting fed. A related PH would be an actual statement of the specific goal that the script is normally assumed to achieve or one from which that goal could easily be inferred. In "John wanted to eat a hamburger" or "John wanted some Italian food," the inference chain to the script precondition is relatively straightforward.

A second type of header, which makes stronger predictions than a PH about the associated context, is the instrumental header (IH). An IH comes up in inputs that refer to two or more scripts, at least one of which can be interpreted as an "instrument" for the others. For example, in "John took the subway to the restaurant," both \$SUBWAY and \$RESTAURANT would be predicted, since subsequent inputs about either would make perfectly good sense. Here, the reference to \$RESTAURANT is anticipatory, and \$SUBWAY is a recognized instrumental means of reaching locales in which more important script goals can be expected to be accomplished.

The notion of a time-place setting for a script leads to the third and most strongly predictive type of header, the locale header (LH). Many organizations have a "residence" or "place of business" in which they characteristically carry on their activities. They may have distinctively designed ornaments or buildings (e.g., a pawn shop's sign, a barber's pole, or McDonald's Golden Arches) that signal their scripts in the world. When an understander reads that an actor is in the proximity of the residence, or, better yet, inside the residence, its expectations about the occurrence of the script are correspondingly reinforced. Examples of LHs are "John went to the soccer field" and "John went into the Museum of Modern Art."

The final type of header is a flat assertion that the script occurred. Examples include:

There was a car accident.  
An earthquake struck.  
John went on vacation.  
Mary went sailing.

Such a direct header (DH) is the top-level pattern in a script. DHs are always the first pattern to be checked in a context, since they have the maximum predictive power.

Here are the headers of the subway script:

Direct header:

```
($SUBWAY
  MAIN  &PATGRP
  PTRORG &SUBORG
  ORIG  &ORIG
  DEST  &DEST)
```

## Locale header:

```
(PTRANS
  Actor    &PATGRP
  OBJECT    &PATGRP
  TO        (INSIDE
             PART &STATION))
```

## Instrumental header:

```
(PTRANS
  ACTOR    &SUBORG
  OBJECT    &PATGRP
  TO        (PROX
             PART &DEST))
```

## Precondition header:

```
(GOAL
  PART      &PATGRP
  OBJECT    (PTRANS
             ACTOR    &PARTORG
             OBJECT    &PARTORG
             TO        (PROX
                         PART &DEST)))
```

The DH is intended to handle conceptualizations corresponding to input such as "John took a subway ride to Coney Island." The LH takes care of sentences such as "John walked into the Boro Hall subway station." The IH will handle conceptualizations such as "The IRT (Interborough Rapid Transit) took John to Shea Stadium." Finally, the PH would match conceptualizations for sentences such as "John wanted to go downtown."

**Script Events, Episodes, and Scenes.** Activities in scripts are stereotyped. Events follow one another in one of a small set of recognized ways. On entering the subway, e.g., the patron may either proceed directly to the turnstile or stop to buy a token. A chain of event patterns describing one of these well-understood activities is called an episode. "Buying a token" is an episode consisting of the events "enter the station," "see the cashier's cage," "go to it," "ask for a token," "be told the fare," and "pay the fare." Each of these events is represented in a language-independent way using conceptual dependency. Note that the script demands that the fare be paid before the token is handed over. This is how the episode is always structured in the subway script, although the actions can be reversed in other scripts, such as when a person is buying an ice cream cone.

Every episode has a main conceptualization, or Maincon, which is the goal or point of the episode. The episodes (marked with "E") and Maincons (marked with "M") of \$SUBWAY are shown in the Figure 1.

In Figure 1 the branching paths at 1 lead to the subsequent episodes E2 and E3, which describe alternative ways of arranging to get through the turnstile. The "loops" at point 2 and 3 are for the cyclic episodes E5 and E7, i.e., for episodes that may happen several times in succession. At 2 several trains may arrive before the one the patron wants, and so the patron must continue to wait.

One could make a single episode out of all the events from entering the subway to leaving the destination station. However, one would clearly be ignoring important facts about the structure of \$SUBWAY. Some parts of the subway ride are more important, more central to the situation, than others.

Getting a token, e.g., is an important activity in \$SUBWAY because without one a patron cannot get through the turnstile to get his ride. Taking a seat on the platform, on the other hand, is not so important because it does not really have any effect on whether the patron can get on the train. Also, sometimes different ways of doing the same thing may be available. Having a token before the ride, asking for one at the counter, or showing the cashier a special pass are all possible ways of procuring a ride. In this case one has a set of episodes that seem to go together.

The activities of a script that always have to occur in order to recognize that the script has in fact been instantiated are called scenes. The scenes of \$SUBWAY are:

\$SUBWAYENTER	Enter the station and wait at the platform
\$SUBWAYRIDE	Enter the train and ride to destination
\$SUBWAYEXIT	Leave the station at destination

"Entering," including buying and using a token, is a scene of the subway script because it is necessary to procure a ride. "Riding" is a scene because this transporting activity is what the script is all about. "Leaving" is a scene because one cannot be sure that the ride is over until the patron exits the station (i.e., the patron might just otherwise be transferring between subway lines). Each scene of a script is defined by a set of episodes that describes the different ways in which an important activity of the scene can happen and other, less important actions, such as sitting down on the platform, that can be interlinked with the main episodes but do not contribute directly to their accomplishment. Scenes commonly organize events that occur within the same sublocale. For example, all the events in the SUBWAYRIDE scene occur in the train locale.

### Why Scripts are Useful

Early attempts to program computers to understand natural language (see Natural-language understanding), despite the initial optimism of the researchers, met with only limited success. For example, the problem of machine translation (qv) between languages was viewed as being essentially one of supplying the computer with dictionaries and grammars of sufficient high quality. The ultimate failure of the translation projects of the 1950s, as described, e.g., by Bar-Hillel (6), can be directly attributed to the reliance on formal, syntactic methods. Knowledge of the world was completely lacking in the systems actually built at that time.

Two of the earliest semantics-based approaches to natural-language processing were represented by the SHRDLU (qv) (7) and MARGIE (qv) (5,54) systems. SHRDLU engaged in an interactive dialogue concerning a microworld of blocks. It could answer questions about blocks and carry out simple commands, such as "Pick up the red pyramid on a blue block," SHRDLU represented the meaning of a sentence in terms of a procedure to carry out a set of actions within the blocks microworld. For instance, "the pyramid red on a blue block" would be translated into a program to examine each block until a blue one was found and then check to see if a red pyramid was on top of it and, if not, continue searching for another blue block, etc. Unfortunately, this approach of procedural representation made it difficult to represent the meaning of a sentence outside the context of a prespecified microworld.

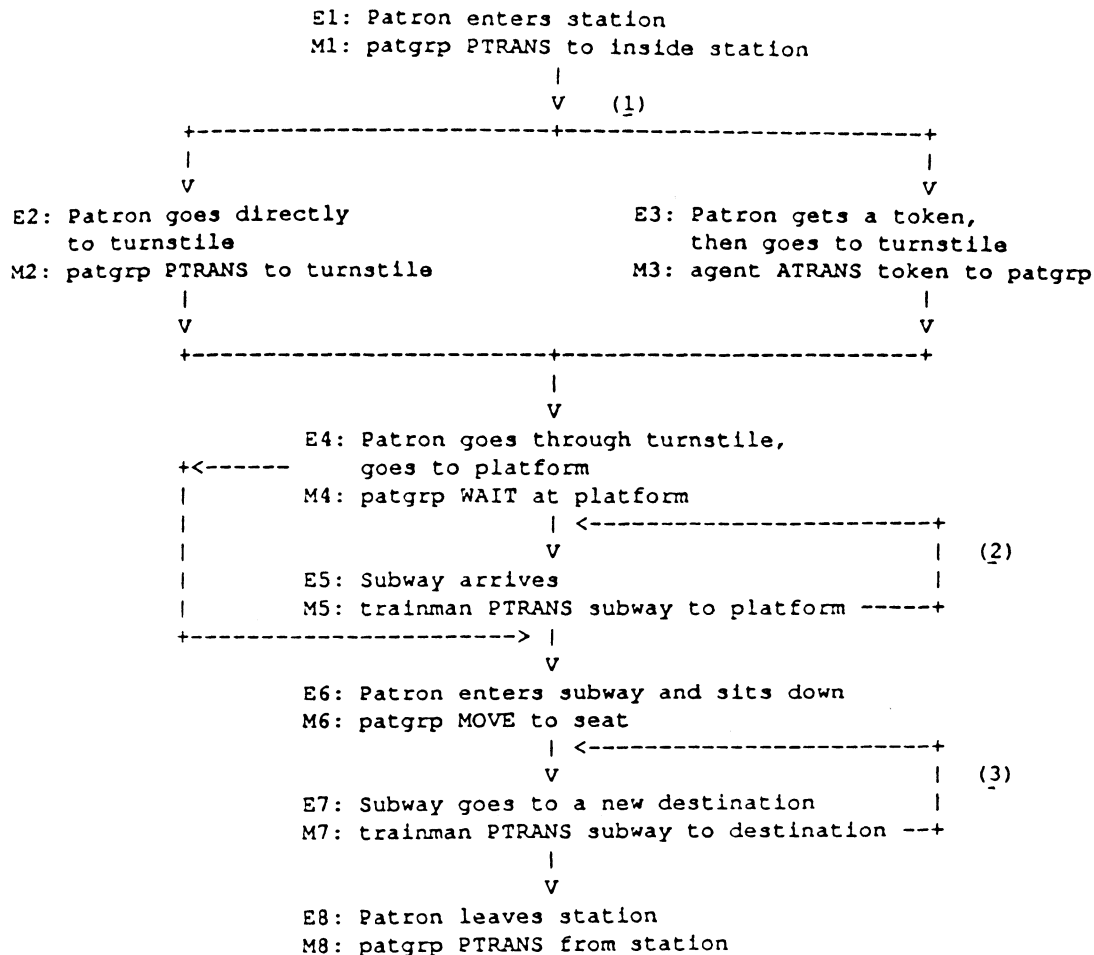


Figure 1. Episodes of subway script.

In contrast, MARGIE was based on a representational system intended to be independent of any particular microworld. MARGIE's representational system was based on a fixed set of CD primitives (4,5) and was composed of three modules: ELI (8), MEMORY (9), and BABEL (10). ELI parsed English sentences, producing CD representations. MEMORY then generated all the inferences that arose from the conceptual representations produced by ELI. These inferences were themselves represented in CD. Finally, BABEL generated paraphrases by expressing each CD conceptualization in English.

However, MARGIE had great difficulty handling more than one sentence at a time. This occurred because MEMORY generated every possible inference it could. Since each inference would generate a CD and since that CD had many potential inferences associated with it, very quickly MEMORY would be overcome by a combinatorial explosion of inferences. There seemed to be no way of constraining inferences to those that were relevant to the text at hand.

The combinatorial problem was partially solved by the notion of a script. Although independently developed, the notion of the script takes the same standpoint as the more general notion of a frame (11,12) (see Frame Theory), which was developed originally to handle problems in vision (qv) (13,14). Like frames, scripts seek to represent a variety of knowledge domains in terms of a hierarchical data structure containing slots that describe what features of the input can bind to these

slots and how to construct default values if these features are missing from the input.

Scripts are useful in supplying expectations during text processing. These expectations represent an active context and help in such tasks as:

1. *Pronoun resolution*: In a \$RESTAURANT context, clearly the "he" in "he left him a big tip" is the customer while "him" must be the waiter.
2. *Word-sense disambiguation*: The expressions "ordered" and "to go" have different meanings in a restaurant ("John ordered a pizza to go") than in the military ("The general ordered a private to go").
3. *Supplying inferences*: Once a script is chosen, processing can proceed very efficiently since missing information is automatically provided by the script. For example, the reader can infer that a waiter or waitress brought the food to the diner even if the text only states "When the food came, he ate it and left a big tip."

Scripts also contain one or more paths (2), each supplying a possible alternative sequence of actions to the "ghost" (or default) path (15). An alternative path in \$RESTAURANT includes leaving without paying because the food was improperly cooked. This path information is used to answer questions, such as:

Q: Why didn't John eat the hamburger?

In this case the search heuristic starts with the "ghost path" in which the food would have been eaten. The retrieval heuristic backs up along this path until a branch point is found. Here resides the reason perhaps:

A: The hamburger was burnt.

that an alternative path was taken.

### Problems with Scripts

A strictly script-based approach to natural-language processing has serious limitations for three reasons:

1. Scripts were conceived as self-contained "chunks" of knowledge. As a result, it is difficult to share knowledge across scripts. For example, a restaurant serves meals, but people also eat meals in nonrestaurant situations (e.g., home and picnics). This meal knowledge should be shared with restaurant knowledge even though the meal server in a restaurant differs from the meal server at home.
2. Experiences occurring within one script cannot be generalized to other relevant situations because scripts are self-contained (16). For example, the knowledge that one can refuse to pay for burnt food in a restaurant should be available in other scriptal contexts. If a mechanic fails to properly fix one's car engine, refusing to pay should come to mind as a potential course of action (17). But this is impossible with scripts since refusing payment is firmly attached to \$RESTAURANT (and not to \$AUTO-REPAIR).
3. Scripts lack intentionality. From a scriptal point of view, each event occurs next simply because it is the next event in the script. Although script-based programs know that characters initiate \$RESTAURANT to satisfy hunger, they do not know why any specific event within \$RESTAURANT occurs. This is analogous to answering:

Q: Why does the diner tip the waitress?

A: I do not know. That is just what he does in a restaurant after he has eaten.

Lack of intentionality has both advantages and disadvantages. It is certainly more efficient, since goals and plans do not have to be processed in order to predict what a character will do next. However, it is difficult to handle novel situations, where a character's reaction might be explainable if the underlying goals and motivations (for the expected event) were known (18,19).

In response to limitations 1 and 2 above, and to experiments (20) regarding memory confusions not predicted by scripts, Schank (21) developed a theory of MOPs (qv) (memory-organization packets). Aspects of this theory were subsequently implemented in a number of text-processing systems. MOPs are memory structures that, like scripts, encode expectations, but unlike scripts, MOPs are not isolated chunks of knowledge. Instead each MOP has strands that indicate how the MOP has been constructed from other knowledge sources. Each strand connects an event in one MOP to some event in another MOP. In this way, MOPs are overlaid on one another. Consider Figure 2, which shows how restaurant knowledge is encoded as MOPs in the BORIS understanding system (22).

Events in M-RESTAURANT are overlaid with their corresponding events in M-MEAL and M-SERVICE. In this way M-RESTAURANT can be viewed from different perspectives. From the perspective of M-MEAL, a restaurant is simply a setting in which people have meals. From the perspective of M-SERVICE, the diner in M-RESTAURANT is engaged in a service contract with the restaurant owner. The restaurant must serve food to the diner and, in return, the diner is expected to pay for this service.

As stated earlier, script-based systems represent scriptal deviations in terms of alternative "paths." Thus, \$RESTAURANT contains BURNED-FOOD→LEAVE-WITHOUT-PAYING path. But this approach causes a proliferation of paths for two reasons:

1. Every possible deviation has to be anticipated and "canned" into the script; otherwise the script would not be able to handle the deviation.
2. Each service-related script ends up copying the same path. For example, in \$AUTO-REPAIR there has to be a BAD-REPAIR→LEAVE-WITHOUT-PAYING path. By overlaying these scripts with M-SERVICE, a single deviation path, POOR-SERVICE→REFUSE-TO-PAY, in M-SERVICE can represent deviation knowledge at a more general level. Likewise, "payment for service" occurs in many situations, not just in \$RESTAURANT. Therefore, tipping should be understood at a more general level than \$RESTAURANT.

When stereotypic situations are first encountered, people let their scriptal knowledge handle the situation until a violation occurs. At this point they become aware of other perspectives associated with the script, each potentially useful in understanding the violation.

For instance, one rarely thinks about the contractual aspects of restaurants when eating in one of them. Restaurants are usually "taken for granted" as a place where one can have a meal and socialize. Only when something goes wrong does one think of the contract one is implicitly engaged in. If the service is bad, one considers the amount of the tip and what it is for.

More important, a deviation that has never before been encountered may be handled as long as some strand exists from the script to a knowledge structure with information about this type of deviation. For instance, if \$MOVIE has strands to M-SERVICE, the very first time the movie projector breaks, one can use the deviation path in M-SERVICE to think of demanding a refund even if the projector breaking down is a novel experience.

### Natural-Language-Understanding Systems Using Scripts

The notion of script has been used by several natural-language-understanding (qv) systems, which include SAM, FRUMP, BORIS, and OpEd. How these systems apply the notion of script during text understanding is described below.

**Script Application in SAM.** SAM (qv) (Script-Applier Mechanism) (2,3,23,24) is a system of computer programs written to investigate how knowledge of context can be used to aid in understanding stories. The basic knowledge source SAM applies is the script. Using scripts of various degrees of complexity, SAM can read (by the process of script application) not





script application. An example of a causal-chain completion inference is to be seen in the answer to question 1, which asks about an implicit, yet normative, event.

Question 4 also shows SAM's ability to analyze in context. The story input "when the hamburger came" has been identified as referring to the "serving" event. Here a role-merging inference has identified "the waiter" as the active entity in this event, given that hamburgers obviously cannot move by themselves.

In addition to inferring what happened while reading the story, SAM is also capable of making inferences at the time of question answering. This arises with questions about events that did not happen. In question 3 the nonoccurrence of the highly expected "eating" event is attributed to an "interfering" condition, i.e., the restaurant was out of hot dogs. Similarly, in questions 5 and 6 SAM infers that the reason for the second nonoccurrence of "eating," followed by the customer's refusal to pay the check, is a second "interfering" condition, i.e., the hamburger was burnt.

Basically, SAM is configured as a set of three modules: ELI (qv) (8), which analyzes the text into a conceptual-dependency (CD) meaning representation; PP-MEMORY, which tags and identifies references to physical objects, and APPLIER, which applies scripts. PP here stands for "picture producer," i.e., anything that is concrete enough to create an image in one's mind. ELI is the only module of SAM that is concerned with linguistic knowledge, and it is the only one that worries about the particular ways that English-speaking people indicate meaning via the choice of word senses, the order of words, the inflection of words, etc. ELI's job in SAM is to extract from an English sentence only the conceptual elements that are there explicitly, avoiding inference as far as possible. In SAM the task of inferring the things an input leaves out is reserved for the "memory" routines (PP-MEMORY and the script applier). This is because making inferences of this type depends on the use of world knowledge rather than on the superficial semantic information ELI possesses as part of its knowledge of English.

When PP-MEMORY is finished processing ELI's output, it sends the result to the script applier. This program has three fundamental problems to solve as it processes a new conceptualization: locating a new input in its database of scripts, setting up predictors about likely inputs to follow, and instantiating the appropriate segment of the script up to the point referred by the input. Of these three problems, the first one, called the script-management problem, is the most important.

The script applier controls the comprehension process by consulting its collection of scripts. Each script has several important parts. First, there are the script's characteristic event chains or episodes. Since an event in a script may be realized in the world in many ways, the events in script episodes are patterns.

A special set of patterns is found in the script "preconditions," or those global facts SAM assumes to be true when a script is entered for the first time unless it reads something to the contrary. When \$RESTAURANT is activated, e.g., the script applier will assert that the patron is hungry and has money to pay for the meal. If the text has indicated that the patron does not have any money (if, e.g., he left his wallet at home), the violation of the precondition will trigger a prediction that the patron will have trouble when it comes time to pay the bill.

An important part of each script is the information that is

always active in memory. This includes static data such as an initial list of those patterns that activate the script (the headers), how related episodes are combined into chunks or "scenes," time and place-setting data for the script, and how other, simpler scripts may be used as units in the main script.

The script applier's three main procedures are a pattern matcher, a predictor, and an instantiator. All three procedures run under an executive, and all have access to script data. The pattern matcher consists of a routine that sets up desired script contexts one at a time, the matcher proper, and a set of auxiliary inference processes. The predictor adds and removes event patterns based on the pattern currently active and what has gone before.

The script applier attempts to understand a story by introducing the most inclusive script it possesses that is initiated by the first conceptualization in the story. As each input conceptualization is recognized in the script, predictions are made by the script applier about future inputs. This cycle continues until the system receives an input that does not refer to a predicted event. At this point it again brings in the largest script the input initiates, matches roles and props across the script interfaces, checks the preconditions, if any, for the new script, and starts matching inputs in the new context.

In the course of story understanding the script applier maintains data structures that describe the state of each script present in the system. This information forms a script context which, if the script is active, is updated whenever a new conceptualization is found to fit within the context. Each script context is defined by the list of patterns from that script that are currently in memory; an association list of tokens bound to script variables; the name of the last pattern matched in the script; the list of script episodes currently in memory; the header for this incarnation of the script; and a script-global inference-strength indicator the applier uses to flag how probable its inferences appear to be.

The most important data structure is the conceptual story representation being constructed by the script applier for the current text. This structure provides access to the final record of the story and is used by the question-answering routines.

The script applier's basic cycle is to call the script contexts one at a time and to attempt to locate an input in the context invoked. Candidate scripts are brought into active memory in the following order: first are those script contexts explicitly referred by the input or indirectly accessed via a PP or a sub-conceptualization in the input; next are the currently active scripts; last are the scripts the system possesses but that have not been invoked.

Once an input has been located in a script context, the Instantiator links it up with what has gone before in that context and then checks on the effect this may have on other active contexts. If the script is being referenced for the first time, the applier checks on the script preconditions to see whether the script is being entered normally or if some unusual events are to be expected in the new context because of previous events. If more than one context is current, the applier may be able to update the story representation on the basis of the static information that is always available for the scripts. For example, \$TRAIN contains the information that a reference to the \$RESTAURANT context via "dining car" in an existing \$TRAIN context defines a "parallel-nested" relationship. Inputs to follow may refer to either script, but \$RESTAURANT should be completed before \$TRAIN.

Many transitions between component scripts are handled

by the more complex scripts that define the "global" context of the story. For example, \$BUS, \$TRAIN, \$PLANE, etc., are known to be "instrumental" means of reaching or leaving a place where the "goal" activity of the trip takes place. The global \$TRIP script may be explicitly introduced, as in "John went to Miami on a business trip," or implicitly referenced by one of its instruments, as in "John took a train to Miami." Script situations, as these scripts are called, provide the most important machinery for the solution of what is called the script-management problem.

**Skimming in FRUMP.** FRUMP (qv) (Fast Reading, Understanding, and Memory Program) (25–27) is a text-understanding program that skims and extracts sparse summaries of news articles. FRUMP displays its understanding by generating a summary that represents the "gist" of a story. Unlike SAM, FRUMP ignores many words in the text. The resulting memory of a story often misses events and situations that occur within the story.

FRUMP reads numerous stories on the UPI (United Press International) news wire. FRUMP contains over 50 "sketchy" scripts concerning such things as earthquakes, kidnappings, visiting dignitaries, labor strikes, and breaking diplomatic relations.

Unlike SAM, analysis in FRUMP is integrated with other processes to the extent that "sketchy" scripts direct it in a "top-down" manner (see Processing, bottom-up and top-down). Once a knowledge structure is referred to in the text, this structure directs subsequent parsing strategies. When FRUMP finds a story that involves one of its scripts, it uses that single script to guide its analysis of the text. In SAM the ELI parser would first analyze a sentence and produce a CD representation for the conceptual content of that sentence. Then the current active script would be applied to the input CD. For instance, in "he ate it," the "he" would be bound to the diner since it is the role of the diner to eat the food. In contrast, FRUMP is organized around a predictor and a substantiator. Once a sketchy script has been selected, the predictor actually directs the substantiator to find in the text what the predictor is looking for. Thus, there is no prior parse phase (i.e., mapping text to CDs) once a sketchy script is active. For instance, assume the \$FIGHTING sketchy script is already active and the predictor has predicted that SHOOTING will occur. At this point the predictor tells the substantiator to try to find an instance of shooting (i.e., PROPEL object = bullets). The substantiator knows about English and searches in the appropriate place in the text for words that have this PROPEL CD conceptualization associated with them. For example, if the substantiator encounters the word "fire" with the appropriate actors (e.g., "the soldiers did fire at the enemy"), it automatically selects the PROPEL-bullets meaning of "fire" rather than the "flame/combustion" meaning of "fire." Thus, the parser is under the control of memory rather than parsing independently and then presenting the results for script application. The advantage of this integrated approach is that the context can direct the comprehension process.

The following I/O example illustrates the level of comprehension that FRUMP attains.

*MOUNT VERNON, ILL. (UPI)—A small earthquake shook several Southern Illinois counties Monday night, the National Earthquake Information Service in Golden, CO, reported.*

*Spokesman Don Finley said the earthquake measured 3.2 on*

*the Richter scale, "probably not enough to do any damage or cause any injuries." The quake occurred about 7:48 P.M. CST [Central Standard Time] and was centered about 30 miles east of Mount Vernon, Finley said. It was felt in Richland, Clay, Jasper, Effington, and Marion counties.*

*Small earthquakes are common in the area, Finley said.*

SELECTED SKETCHY SCRIPT \$EARTHQUAKE.

CPU TIME FOR UNDERSTANDING = 3040 MILLISECONDS

ENGLISH SUMMARY:

THERE WAS AN EARTHQUAKE IN ILLINOIS WITH A 3.2 RICHTER SCALE READING.

To process the story above, FRUMP uses an earthquake "sketchy" script and extracts the place and magnitude of the quake. Everything else, however, is ignored.

The top-down approach fulfilling prior expectations and selectively ignoring information not conforming to these expectations gives FRUMP a great deal of robustness. The main negative consequence is that unusual or unexpected information is often missed since it does not conform to the predictor's script-based expectations. As a result, unusual (and interesting) events in the text fail to be incorporated into a final memory representation of the story. This approach is adequate, however, for the task of extracting sparse summaries for stories via skimming.

**Integrated In-Depth Parsing in BORIS.** BORIS (qv) (17,22,28–30) is an in-depth story-understanding and question-answering system that deals with the specification and interaction of many sources of knowledge. In contrast to text skimmers, BORIS attempts to understand narratives as deeply as possible. For BORIS understanding a narrative "in-depth" means the following: reading in a careful mode rather than skimming; handling narratives that involve multiple interacting knowledge sources; parsing text in an integrated fashion, where memory search and construction processes are evoked on a word-by-word basis; and recognizing the key thematic patterns that characterize a narrative at very abstract levels.

BORIS attempts to construct a complete representation of a narrative, including all physical events and mental states, along with the causal connections between them. As a result, BORIS operates in a very bottom-up manner. Processing is directed more from information arising in the input than from predetermined expectations as in FRUMP. Expectations in BORIS are encoded in the episodic memory of the narrative read thus far and are activated by search processes. This bottom-up approach gives BORIS the capability of noticing unusual events, which are often missed by exclusively top-down processing approaches. It is the unusual and unexpected events, including the mistakes and failures of the characters, that often make a story memorable. By their very nature, such events cannot be predicted in a top-down manner.

BORIS is a highly integrated system. All memory search, instantiation, and inference tasks occur as side effects of a single, unified parsing process that occurs on a word-by-word basis. That is, BORIS neither parses a complete sentence before calling a script applier nor relies on an active script to tell the parser completely how to map text into conceptualizations. Instead, BORIS associates parsing and memory-search processes with each word (phrase) in its lexicon. As a word or phrase is encountered, these processes (called "demons" (qv))

knit together CD fragments from the input, perform inferences, call upon script applicators, and search memory for a number of different knowledge constructs. Furthermore, narrative questions are parsed by the same processes that handle the narratives themselves. One natural consequence is that BORIS often knows the answer to a question before it has completely understood the question. Another natural consequence is a "Loftus Effect" (31,53). That is, asking a question about a narrative may cause the memory of the narrative to be altered (22).

In contrast to early script-based systems that operate within restricted knowledge-source domains and deal poorly with script interactions, BORIS uses a MOP overlay scheme to achieve a much higher level of generality, which is especially useful in processing norm violations when they occur within scriptal contexts. For example, consider how BORIS processes the following fragment of DIVORCE-2 (22), a complicated narrative concerning marital infidelity:

*George was having lunch . . . when the waitress accidentally knocked a glass of coke on him. George was very annoyed and left refusing to pay the check.*

Q1: Where did George have lunch?

A1: AT THE RESTAURANT.

Q2: What happened at the restaurant?

A2: THE WAITRESS SPILLED COKE ON GEORGE AND HE REFUSED TO PAY THE CHECK.

Q3: How did George feel at the restaurant?

A3: GEORGE WAS ANGRY BECAUSE THE WAITRESS SPILLED COKE ON HIM.

In order to understand this fragment, BORIS uses the MOP overlay scheme shown in Figure 2. Briefly, an analysis of "having lunch" activates M-MEAL. When "waitress" occurs, the MOPs associated with this role are examined. If a MOP being examined has a strand to an active MOP, it is also activated. As stated earlier, there are strands from M-RESTAURANT to M-MEAL. Thus, since M-MEAL is already active, M-RESTAURANT is activated also. (This approach avoids the problem of invoking the restaurant script simply at the mention of a waitress, as in "John is in love with a waitress.")

An interpretation of "accidentally" indicates that a violation may follow. This heuristic is based on the assumption that unintended actions usually violate scriptal expectations.

"Knocked a glass of coke on him" is analyzed in terms of the CD primitive PROPEL with Object = Liquid. Given this event, BORIS tries to match it against the event expected in M-RESTAURANT. This match would normally fail since M-RESTAURANT does not expect waitresses to PROPEL foodstuffs. However, "accidentally" has warned BORIS of a possible violation, so a violation match is attempted and succeeds. At this point BORIS realizes that the PROPEL event is a violation of the event BRING-FOOD in M-RESTAURANT rather than some event totally unrelated to M-RESTAURANT.

Now what is BORIS to do? In previous systems there would be a prespecified path in the script for such a deviation. However, this is not the case here. When BORIS encounters a deviation, it searches the strands connected to the event where the deviation occurred. This leads to DO-SERVICE in M-SERVICE (see Fig. 2).

Associated with M-SERVICE is general knowledge about how things may "go wrong" for each event in M-SERVICE. There are several events, such as ARRANGE-SERVICE, DO-SERVICE, INFORM-BILL, MAKE-PAYMENT, etc. For example, the sentence

The waitress overcharged George

constitutes a violation of INFORM-BILL.

In addition, there is knowledge about how violations may be related to each other. This knowledge is represented by rules, such as:

*IF SERVER has done SERVICE badly (or not at all), THEN SERVER should either not BILL CONTRACTOR or BILL for amount less than NORM.*

*IF SERVER has done service badly or BILLS CONTRACTOR for amount greater than NORM, THEN contractor may refuse PAYMENT.*

BORIS uses this knowledge to recognize the connection between the waitress PROPEL LIQUID and George's refusal to pay a check.

Finally, consider how BORIS realizes that the violation of BRING-FOOD actually constitutes POOR-SERVICE. This is accomplished by tracking the goals of the characters. The PROPEL LIQUID on George is understood to cause a PRE-SERVE-COMFORT goal for George. This goal is examined by M-SERVICE, which applies the following heuristic:

*IF SERVER causes a PRESERVATION GOAL for CONTRACTOR while performing SERVICE, THEN it is probably POOR-SERVICE.*

Thus, BORIS uses several sources of knowledge to understand what has happened. M-RESTAURANT supplies expectations for what the waitress should have done. Knowledge about PROPEL and LIQUIDS supplies goal information, and M-SERVICE (between waitress and diner) provides general knowledge about how contractors will respond to poor service.

The overlay scheme used in BORIS causes equivalences to be set up among component structures in different MOPs and has three major advantages:

1. Each knowledge structure needs to know only what is directly relevant to it. For example, what a waitress does is captured in M-RESTAURANT, although her reasons for doing her job are represented at the M-SERVICE level (which handles any type of service); thus M-SERVICE need not be repeated for janitors, salespersons, etc. This supports economy of storage, but more important, it means that any augmentation of the knowledge in M-SERVICE will automatically improve the processing ability of any MOP with strands to it.
2. Related knowledge sources need not be activated unless something goes wrong during processing. For instance, people do not normally think of the contract between themselves and the restaurant manager unless they are having trouble with the service.
3. A given event can be understood from several perspectives. For example, a "business lunch" involves M-MEAL, M-SERVICE, M-RESTAURANT, and M-BUSINESS-DEAL simultaneously.

**Reasoning Scripts in OpEd.** OpEd (Opinions to/from the Editor) (32) is a computer system that reads short politico-economic editorial segments and answers questions on the editorial contents. OpEd's design is based on the conceptual parser implemented in BORIS, the question-answering theory developed by Dyer and Lehnert (29) as an extension of previous work by Lehnert (15), and the argument graph developed by Flowers et al. (33).

Input editorial segments are in English and contain essential issues and arguments of the original editorials. The memory representation built while parsing an editorial forms a graph composed of five major elements: instantiations of domain-specific objects (e.g., nations), goals, plans, events, states, affective reactions, and participants in editorial arguments; instantiations of argument participants' beliefs; an argument graph that includes all instantiated beliefs, belief relationships, and belief justifications; instantiations of argument units (AUs) that organize abstract knowledge about reasoning and argumentation; and indexing structures that allow search and retrieval processes access to the argument graph during question answering.

Like BORIS, OpEd is a highly integrated system. Memory and inference tasks are performed on a word-by-word basis by active processes called demons. These tasks include interactions of knowledge sources, role binding, word disambiguation, concept references, belief inferences, inference of belief relationships, justification tracing, question categorization, and answer retrieval.

Part of OpEd's causal politico-economic knowledge is represented by reasoning scripts (34), which package causal domain knowledge in the form of prespecified reasoning-chain sequences. OpEd recognizes and instantiates reasoning scripts when it is following belief justifications that contain structural gaps, i.e., justifications involving causal chains with implicit cause-effect relationships. Consider how OpEd processes the following fragment of ED-JOBS (32), a segment from an editorial by Friedman (35):

*Recent protectionist measures by the Reagan administration have disappointed us. . . . Far from saving jobs, the limitations on imports will cost jobs. If we import less, foreign countries will earn fewer dollars. They will have less to spend on American exports. The result will be fewer jobs in export industries.*

Q: Why does Milton Friedman believe that the limitations on imports will cost jobs?

A: MILTON FRIEDMAN BELIEVES THAT PROTECTIONIST POLICIES . . . WILL THWART THE PRESERVATION OF JOBS FOR U.S. BECAUSE . . . IF U.S. IMPORTS FEWER PRODUCTS, THEN THERE IS A DECREASE IN THE PROFITS OF FOREIGN COUNTRIES. IF THERE IS A DECREASE IN THE PROFITS OF FOREIGN COUNTRIES, THEN FOREIGN COUNTRIES BUY FEWER AMERICAN EXPORTS. IF FOREIGN COUNTRIES BUY FEWER AMERICAN EXPORTS, THEN THERE IS A DECREASE IN THE PROFITS OF EXPORT INDUSTRIES. IF THERE IS A DECREASE IN THE PROFITS OF EXPORT INDUSTRIES, THEN THERE IS A DECREASE IN JOBS IN EXPORT INDUSTRIES. A DECREASE IN JOBS IN EXPORT INDUSTRIES THWARTS THE PRESERVATION OF JOBS FOR U.S.

In order to understand Friedman's complex reasoning chain, which justifies his belief that the limitations will cost jobs, OpEd must recognize and instantiate the implicit relationship between a decrease in exports and a decrease in jobs, namely:

*IF COUNTRY C1 spends less on PRODUCT P produced by PRODUCER P1 from COUNTRY C2, THEN there is a decrease on the EARNINGS of PRODUCER P1. AND IF there is a decrease on the EARNINGS of PRODUCER P1, THEN there is a decrease in the number of OCCUPATIONS in the industry group of PRODUCER P1.*

This cause-effect chain is contained in the reasoning script \$R-DROP-FOREIGN-SPENDING→DROP-JOBS. During the instantiation process, C1 is bound to "foreign countries," C2 to "U.S.," and P1 to "U.S. export industries." Thus, instantiating this reasoning script allows OpEd to infer that a decrease in U.S. exports causes a decrease in jobs in U.S. export industries. This inferred relationship is later generated as part of the answer to the question above, as:

*IF FOREIGN COUNTRIES BUY FEWER AMERICAN EXPORTS, THEN THERE IS A DECREASE IN THE PROFITS OF EXPORT INDUSTRIES. IF THERE IS A DECREASE IN THE PROFITS OF EXPORT INDUSTRIES, THEN THERE IS A DECREASE IN JOBS IN EXPORT INDUSTRIES.*

The use of reasoning scripts allows OpEd to infer missing steps in a chain of reasoning that justifies a particular belief in the editorial.

### Psychological Validity of Scripts

The script notion has generated much interest among psychologists. Scripts help computer programs represent and organize their knowledge of stereotypical events. In addition, it seems reasonable to believe that humans use their stereotypical knowledge of the world to understand and behave appropriately in everyday situations, such as going to a supermarket or a restaurant. At this point, consider whether humans organize their stereotypical knowledge in scripts, i.e., whether the concept of script is psychologically valid.

First, it is clear that children acquire cognitive structures exhibiting the major features of scripts very early in their development. In a series of open-ended interviews, e.g., Nelson and Gruendel (36) found that preschool children up to the age of three can reliably report on the ordering of activities and the characteristic roles in common activities such as going to McDonald's or to a birthday party. In fact, children also learn these structures very rapidly. Within a week or so after entering a new situation, such as preschool, they have formed a clear notion of their expected roles, settings, and sequences of activities.

As reported in Ref. 37, a great many experiments have been carried out to investigate basic properties of scripts. For instance, Bower et al. (20) and Graesser et al. (38) have demonstrated that people tend to infer and recognize unmentioned script events when presented with stories about scripted activities. In these experiments subjects were first asked to read script-based stories that contained only some of the normative events of the evoked scripts. Later, they were asked to write

down all actions they could remember from the stories read. Results showed that people cited actions from the scripts that were not explicitly mentioned in the stories but that could be inferred from the script.

Bower et al. (20) have also found that when people are asked to remember script-based stories that present some events displaced from their usual positions, they tend to recall the stories with those events shifted toward their normative positions. Abelson (37) believes that this can be explained as a compromise in reconstructive memory between the known event ordering and the presented ordering, since, in general, subjects can agree on event ordering in scripts (39).

Other experiments by Bower et al. (20) have further shown that understanding a given event in terms of a recognized script does not require searching through a sequential list of the script events until the given event is found. For example, understanding the eating event after processing the ordering event within \$RESTAURANT does not require instantiating the low-level events (e.g., holding the fork) that follow the ordering event and precede the eating event. According to Abelson (37), this can be explained because some script events are more central than others to the script and flow of action within the script and central events summarize script scenes containing low-level events. Moreover, Galambos and Rips (39) have demonstrated that it is faster to verify central events as belonging to the script than low-level events. This 'centrality' concept was originally proposed in Ref. 1 and implemented as script main conceptualizations, or MAINCONS, in Refs. 2 and 3.

A number of experiments have also been done to investigate how knowledge of scriptal activities is used during question answering. Galambos and Black (40) have shown that when answering questions about the reasons for actions in scriptal activities, the most frequently used strategy is to answer with a subsequent action in the script. For example, if within the activity MAKING-A-CAMPFIRE a person is asked for the reason of the action GETTING-MATCHES, the most common answer is the next action in that activity, i.e., LIGHTING-THE-FIRE. In addition to this causality, it has also been shown in Refs. 41-44 that the time to answer questions about stereotypical activities is influenced by four other features:

1. distinctiveness, a measure of whether the action occurs in one or many activities;
2. centrality, the importance of the event to the backbone of the script;
3. standardness, a measure of the frequency with which the action is performed within a given activity; and
4. sequential position of the action within the activity.

For example, people do not have difficulty answering that \$RESTAURANT is the activity in which the action SEE-HEAD-WAITER can be performed since this action appears in few (if any) other activities, i.e., SEE-HEAD-WAITER is highly distinctive to \$RESTAURANT.

Bower et al. (20) have also demonstrated that memories for script episodes are not necessarily stored with the representation for the particular script, but rather they are represented as scenes that can be shared among scripts. This sharedness results in confusions at recall time. For example, the waiting-room scene is shared by the doctor's and dentist's office scripts. Thus, memories of events indexed by the waiting-room scene

may not preserve distinction between the doctor's and dentist's offices.

These memory-confusion results demonstrated that the original notion of a SAM-style script as a self-contained unit of knowledge is too restrictive. In fact, the development of MOPs by Schank and his colleagues (16,21) came about as a response to these experimental results.

Finally, Reiser et al. (45) have performed experiments that give evidence for a memory organization based on general actions, i.e., memory structures that encode generalizations about actions common to stereotypical activities. For instance, the general action MAKE-RESERVATIONS is common to the activities GOING-ON-VACATIONS and PLAYING-IN-DOOR-TENNIS. These general actions correspond to the generalized scenes (16,21) around which MOPs are organized.

### Script Acquisition

It has been shown how scripts allow people to recognize and understand events they have encountered many times. Being able to use scripts provides an efficient way to act in stereotypical situations without having to explain them repeatedly in terms of the plans and goals of the events' characters. One faces many situations that are highly scriptal in nature. If scripts do not already exist to handle them, they must be learned. How new scripts are learned and how they organize episodes already existing in memory are still open-ended problems.

Schank and Abelson (1) believe that children learn about stereotypical activities, such as going to restaurants and department stores, by going through those experiences enough times. Dyer (22) pointed out that children (and adults) can learn how to behave in these situations without having to understand the intentional structures underlying their behavior. For example, suppose a child goes to a restaurant for the first time. Although events relating to HAVING-A-MEAL (i.e., SET-TABLE, BRING-FOOD, SIT, and EAT) would be familiar, he would notice many novel events and would try to process them in terms of HAVING-A-MEAL. For instance, he would notice that it is the waitress who sets the table and brings the food in a restaurant (rather than one of his parents). Furthermore, he would also see his parents leaving money at the table and paying at the cashier. However, his understanding of these last events would simply be "that's what one does" after eating in a restaurant. These events are harder to understand in terms of HAVING-A-MEAL because they do not have analogous counterparts. It might be years before he realizes the full contractual significance of tipping and paying the bill. For instance, a friend admitted to Dyer that as a child, she enjoyed pocketing the coins people left on restaurant tables. One day she was caught by her parents, who informed her that these coins were to reward the waitress for her service. Until this moment, in which this child grasped the intentional significance of tipping, it had just been a "scriptal action" for her.

DeJong (46) has outlined an approach to learning schemas (i.e., any knowledge chunks, such as a script, frame, or MOP) called explanatory schema acquisition. In this approach understanding an event for which there is no previous schema requires generalizing the new event into a new schema. This generalization process constructs an explanation of relationships among components of the new event by applying knowledge about the underlying goals and plans involved in the

event. The generalization process distinguishes four types of situations:

1. schema composition, connecting known schemas in a novel way;
2. schema alteration, modifying a nearly correct schema so that it fits the requirements of a new situation;
3. secondary-effect elevation, acquiring a new schema that is nearly the same as an existing schema but whose main effect is only a side effect in the original schema; and
4. volitionalization, transforming a schema for which there is no planner into a schema which can be used by a planner to attain a specific goal.

For example, learning about kidnapping, i.e., building a KIDNAP schema, involves combining two previous known schemas: THEFT (in this case of a person) followed by BARGAIN (in this case the paying for the well-being of the victim). An example of schema alteration involves the case in which \$RESTAURANT is modified to account for new situations in which the typical scene ordering is altered, such as the first visit to a fast-food restaurant where the patron has to pay before eating the food (16).

Schema acquisition can also be accomplished by specializing previously learned schemas. This approach has been used in IPP (47,48), CYRUS (49–51), and OCCAM (52). IPP (Integrated Partial Parser) is a computer system designed to read and form generalizations from a large number of news stories about terrorism. IPP's knowledge about terrorism is organized by two specific types of knowledge structures (47): action units, which describe events, such as shootings, deaths, and bombings, and simple MOPs (S-MOPs), which organize abstract levels of actions, such as extortions and attacks on people. IPP maintains a long-term episodic memory of stories it has read and uses this memory as a basis for making generalizations about terrorism. New generalizations are indexed by new S-MOPs that are specializations of existing ones. For instance, if IPP reads several stories about attacks against the British by the IRA, IPP will make the generalization that terrorist attacks in Britain are normally caused by the IRA. IPP can use this generalization to infer terrorists' affiliation if it is not mentioned in later stories involving terrorism in Britain.

CYRUS (Computerized Yale Retrieval and Updating System) is a computer program that organizes and searches a model of dynamic memory based on episodes from the lives of former Secretaries of State Cyrus Vance and Edmund Muskie. Episodes in CYRUS are organized by Episodic MOPs (E-MOPs) (49,50). E-MOPs organize similar events with respect to each other by indexing them according to their differences. For example, the "diplomatic meeting" E-MOP indexes each of Vance's or Muskie's diplomatic meetings. In CYRUS creation of new E-MOPs that hold generalizations about events is triggered by reminders (16). Reminders occur when CYRUS indexes a second event where a first is already indexed. In this case similarities between the two events are extracted, and a new E-MOP with generalized information based on those occurrences is created. For example, the first time CYRUS hears about a meeting in which Vance discusses military aid with a foreign defense minister, it indexes that meeting uniquely in the "diplomatic meetings" E-MOP under the property "underlying topic is military aid." The second

time Vance meets about military aid, CYRUS is reminded of the first episode because both have the same topic. It checks the descriptions of both, and if the second one involves a defense minister, CYRUS concludes that "meetings about military aid are usually with defense ministers." Then CYRUS creates a new "meetings about military aid" E-MOP and indexes the two meetings within that E-MOP. Although the memory-organization and -generalization processes in IPP and CYRUS are similar, IPP's emphasis is on memory-based parsing whereas CYRUS's is on episodic memory retrieval and reconstruction.

OCCAM is a computer program that organizes memories of events and learns by creating explanatory and tentative generalized events. OCCAM distinguishes three types of generalized events (52): explanatory generalized events, MOPs created as specialization of a more general MOP; tentative generalized events, MOPs inductively generalized from examples without an appropriate general MOP; and organizational generalized events, which correspond to the factual generalizations of IPP involving S-MOPs. In contrast to DeJong's explanatory schema acquisition (46), OCCAM can learn incrementally since there are cases in which a specialized new schema cannot be created from the first example encountered. For instance, the motivation for kidnapping infants (i.e., they cannot testify against the kidnapper) cannot be learned from the first example of a kidnapping since the explanation process can find an explanation for kidnapping any person. In OCCAM, after the basic KIDNAP schema is learned, later examples direct OCCAM to explain coincidences about the age of the victims and make the appropriate generalization.

## Conclusions

The authors have discussed the notion of a script, a memory structure that organizes chunks of stereotypical cultural knowledge. The script concept has created much interest among natural-language researchers in both psychology and AI. This construct has proved useful:

- in theories of representation and organization of knowledge, where scripts capture the stereotypic, cultural knowledge needed by natural-language-understanding systems to process script-based texts;
- in control of inferences, where scripts reduce combinatorial problems by containing the most relevant, normative event sequences as a memory structure rather than having to rederive event relationships through general problem solving and planning each time;
- in psychological models of human narrative comprehension, where scripts explain human behavior in stereotypical situations (37); and
- in language analysis, where scripts help resolve pronoun references and perform word-sense disambiguation when dealing with script-based text.

The notion of script has probably shed light on some of the basic problems any intelligent system must address: knowledge representation, organization, acquisition, and application. Furthermore, the notion of script has contributed to bringing together researchers from AI and psychology in their quest for efficient and accurate models of human cognition.



## BIBLIOGRAPHY

1. R. C. Schank and R. P. Abelson, *Scripts, Plans, Goals, and Understanding*, Erlbaum, Hillsdale, NJ, 1977.
2. R. E. Cullingford, Script Application: Computer Understanding of Newspaper Stories, Ph.D. Thesis, Research Report No. 116, Department of Computer Science, Yale University, New Haven, CT, 1978.
3. R. E. Cullingford, SAM, in R. C. Schank and C. K. Reisbeck (eds.), *Inside Computer Understanding: Five Programs Plus Miniatures*, Erlbaum, Hillsdale, NJ, 1981, pp. 75–119.
4. R. C. Schank, Identification of Conceptualizations Underlying Natural Language, in R. C. Schank and K. M. Colby (eds.), *Computer Models of Thought and Language*, W. H. Freeman, San Francisco, CA, 1973, pp. 184–247.
5. R. C. Schank (ed.), *Conceptual Information Processing*, North-Holland, New York, 1975.
6. J. Bar-Hillel, *Language and Information*, Addison-Wesley, Reading, MA, 1964.
7. T. Winograd, *Understanding Natural Language*, Academic Press, New York, 1972.
8. C. K. Riesbeck, Conceptual Analysis, in R. C. Schank (ed.), *Conceptual Information Processing*, North-Holland, New York, 1975, pp. 83–156.
9. C. J. Rieger III, Conceptual Memory and Inference, in R. C. Schank (ed.), *Conceptual Information Processing*, North-Holland, New York, 1975, pp. 157–288.
10. N. Goldman, Conceptual Generation, in R. C. Schank (ed.), *Conceptual Information Processing*, North-Holland, New York, 1975, pp. 289–371.
11. E. Charniak, "A framed painting: Representation of a common-sense knowledge fragment," *Cog. Sci.* 1(4), 355–394 (1977).
12. E. Charniak, "On the use of framed knowledge in language comprehension," *Artif. Intell.* 11(3), 225–265 (1978).
13. M. A. Minsky, Framework for Representing Knowledge, in P. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.
14. M. A. Minsky, Frame-System Theory, in P. Johnson-Laird and P. Wason (eds.), *Thinking: Readings in Cognitive Science*, MIT Press, Cambridge, Mass, 1977.
15. W. G. Lehnert, *The Process of Question Answering*, Erlbaum, Hillsdale, NJ, 1978.
16. R. C. Schank, *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*, Cambridge University Press, New York, 1982.
17. M. G. Dyer, \$RESTAURANT Revisited or "Lunch with BORIS," *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., 1981.
18. R. Wilensky, Understanding Goal-Based Stories, Ph.D. Thesis, Research Report No. 140, Department of Computer Science, Yale University, New Haven, CT, 1978.
19. R. Wilensky, *Planning and Understanding: A Computational Approach to Human Reasoning*, Addison-Wesley, Reading, MA, 1983.
20. G. H. Bower, J. B. Black, and T. J. Turner, "Scripts in memory for text," *Cog. Psychol.* 11, 177–220 (1979).
21. R. C. Schank, Reminding and Memory Organization: An Introduction to MOPs, in W. G. Lehnert and M. H. Ringle (eds.), *Strategies for Natural Language Understanding*, Erlbaum, Hillsdale, NJ, 1982, pp. 455–493.
22. M. G. Dyer, *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension*, MIT Press, Cambridge, MA, 1983.
23. R. E. Cullingford, "An approach to the representation of mundane world knowledge: The generation and managements of situational scripts," *Am. J. Computat. Ling.*, Microfiche 44 (1975).
24. R. E. Cullingford, The Uses of World Knowledge in Text Understanding, *Proceedings of the Sixth International Conference on Computational Linguistics*, Ottawa, Canada, 1976.
25. G. F. DeJong II, "Prediction and substantiation: A new approach to natural language processing," *Cog. Sci.* 3, 251–273 (1979).
26. G. F. DeJong II, Skimming Stories in Real Time: An Experiment in Integrated Understanding, Ph.D. Thesis, Research Report No. 158, Department of Computer Science, Yale University, New Haven, CT, 1979.
27. G. F. DeJong II, An Overview of the FRUMP System, in W. G. Lehnert and M. H. Ringle (eds.), *Strategies for Natural Language Understanding*, Erlbaum, Hillsdale, NJ, 1982, pp. 149–176.
28. M. G. Dyer, Integration, Unification, Reconstruction, Modification: An Eternal Parsing Braid, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., 1981.
29. M. G. Dyer and W. G. Lehnert, Question Answering for Narrative Memory, in J. F. Le Ny and W. Kintsch (eds.), *Language and Comprehension*, North-Holland, Amsterdam, 1982, pp. 339–358.
30. W. G. Lehnert, M. G. Dyer, P. N. Johnson, C. J. Yang, and S. Harley, "BORIS: An in-depth understander of narratives," *Artif. Intell.* 20(1), 15–62 (1983).
31. E. F. Loftus, *Eyewitness Testimony*, Harvard University Press, Cambridge, MA, 1979.
32. S. J. Alvarado, M. G. Dyer, and M. Flowers, Memory Representation and Retrieval for Editorial Comprehension, *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, Irvine, CA, 1985.
33. M. Flowers, R. McGuire, and L. Birnbaum, Adversary Arguments and the Logic of Personal Attacks, in W. G. Lehnert and M. G. Ringle (eds.), *Strategies for Natural Language Understanding*, Erlbaum, Hillsdale, NJ, 1982, pp. 275–294.
34. M. Flowers and M. G. Dyer, Really Arguing with your Computer in Natural Language, *Proceedings of the National Computer Conference*, 1984.
35. M. Friedman, "Protection that hurts" (Editorial), *Newsweek* 90 (November 15, 1982).
36. K. Nelson and J. Gruendel, Generalized Event Representations: Basic Building Blocks of Cognitive Development, in A. Brown and M. Lamb (eds.), *Advances in Developmental Psychology*, Vol. 1, Erlbaum, Hillsdale, NJ, 1981.
37. R. P. Abelson, "The psychological status of script," *Am. Psychol.* 36, 715–729 (1981).
38. A. C. Graesser, S. B. Woll, D. J. Kowalski, and D. A. Smith, "Memory for typical and atypical actions in scripted activities," *J. Exper. Psychol. Hum. Learn. Mem.* 6, 503–515 (1980).
39. J. A. Galambos and L. J. Rips, The Representation of Events in Memory, Paper presented to the Midwestern Psychological Association, 1979.
40. J. A. Galambos and J. B. Black, Using Knowledge of Activities to Understand and Answer Questions, in A. C. Graesser and J. B. Black, *The Psychology of Questions*, Erlbaum, Hillsdale, NJ, 1985.
41. J. A. Galambos, "Normative studies of six characteristics of our knowledge of common activities," *Behav. Meth. Instru.* 15, 327–340 (1983).
42. J. A. Galambos and J. B. Black, Getting and Using Context: Functional Constraints on the Organization of Knowledge, *Proceedings of the Fourth Conference of the Cognitive Science Society*, Ann Arbor, MI, pp. 44–46, 1982.

43. J. A. Galambos and L. J. Rips, "Memory for routines," *J. Verb. Learn. Verb. Behav.* **21**, 260–281 (1982).
44. B. J. Reiser, J. A. Galambos, and J. B. Black, Retrieval from Semantic and Autobiographic Memories, Paper presented at the Twenty-third Annual Meeting of the Psychonomic Society, 1982.
45. B. J. Reiser, J. B. Black, and R. P. Abelson, "Knowledge structures in the organization and retrieval of autobiographical memories," *Cog. Psychol.* **17**(1), 89–137 (1985).
46. G. F. DeJong II, Automatic Schema Acquisition in a Natural Language Environment, *Proceedings of the Second National Conference on Artificial Intelligence*, Pittsburgh, PA, pp. 410–413, 1982.
47. M. Lebowitz, Generalization and Memory in an Integrated Understanding System, Ph.D. Thesis, Research Report No. 186, Department of Computer Science, Yale University, New Haven, CT, 1980.
48. M. Lebowitz, "Memory-based parsing," *Artif. Intell.* **21**(4), 363–404 (1983).
49. J. L. Kolodner, "Maintaining organization in a dynamic long-term memory," *Cog. Sci.* **7**(4), 243–280 (1983).
50. J. L. Kolodner, "Reconstructive memory: A computer model," *Cog. Sci.* **7**(4), 281–328 (1983).
51. J. L. Kolodner, *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model*, Erlbaum, Hillsdale, NJ, 1984.
52. M. J. Pazzani, Explanation and Generalization Based Memory, *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, Irvine, CA, pp. 323–328, 1985.
53. E. F. Loftus, "Leading questions and the eyewitness report," *Cog. Psychol.* **7**, 560–572 (1975).
54. R. C. Schank, N. Goldman, C. J. Rieger III, and C. K. Riesbeck, "Inference and paraphrase by computer," *J. Assoc. Comput. Mach.* **22**(3), 309–328 (1975).

M. DYER  
UCLA

R. CULLINGFORD  
Georgia Institute of Technology

S. ALVARADO  
UCLA

## SEARCH

Search is a universal problem-solving mechanism in AI (see Problem solving). In AI problems, almost by definition, the sequence of actions required for solution are not known a priori but must be determined by a systematic trial-and-error exploration of alternatives. All that is required to formulate a search problem is a set of states of the world, a set of operators that change the state, an initial state, and a set of goal states. The task then becomes one of finding a sequence of operators that transform the initial state into a goal state. In general, more than one operator can be applied to a given state, which gives rise to search behavior.

Almost every AI program contains a search algorithm, often as the critical inner loop. Typical search tasks range from solving puzzles like Rubik's Cube (Ideal Toy Co.), to playing games such as chess (see Computer chess methods), to solving complex real-world problems such as medical diagnosis (see Medical-advice systems). The spectrum of search algorithms extends from brute-force techniques, which use no knowledge of the problem domain, to knowledge-intensive heuristic searches (see Heuristics).

This brief exposition begins by describing the different problem spaces in which search takes place, primarily state

spaces and problem-reduction spaces. Brute-force searches are then considered including breadth-first, depth-first, depth-first iterative-deepening, and bidirectional search. Finally, various heuristic searches are examined including hill-climbing, best-first search, the  $A^*$  algorithm, beam search, and iterative-deepening- $A^*$ . The efficiency of these algorithms in terms of the lengths of the solutions they generate, the amount of time the algorithms take to execute, and the amount of computer memory they require are of a central concern throughout. The reason for this focus is that since search is a universal problem-solving method, the only thing that limits its applicability is the efficiency with which it can be performed.

## Problem Spaces

A problem space is the environment in which a search takes place (1). A problem space consists of a set of states of the problem and a set of operators that change the state of the problem. For example, in a problem such as Rubik's Cube, the states are the different possible configurations of the puzzle, and the operators are the different physical twists that change its state. A problem instance is a problem space together with an initial state and a goal state. In the case of Rubik's Cube the initial state would be whatever (e.g., random) initial configuration the puzzle starts out in, and the goal state is typically the one in which all sides have a uniform color. The problem-solving task is to find a sequence of operators that map the initial state into the goal state.

In Rubik's Cube the goal state is given explicitly. In other problems, however, the goal state is not given explicitly but rather is specified by certain properties that must be satisfied by any goal state. For example, the Eight-Queens problem is to place eight queens on a chessboard so that no two queens are attacking each other along a row, column, or diagonal. The task here is not to find a sequence of actions but rather to exhibit a particular state that satisfies the goal criterion. Since this problem is too difficult to be solved in a single step, it must be broken down into a series of simpler steps. This typically involves defining states that are partial assignments of queens to board positions, and operators that consist of placing an additional queen on the board. Thus, even though the sequence of operators required to solve the problem is not of interest, the problem is still formulated as a search through a problem space.

**Graph Representation.** In order to abstract away irrelevant details of a particular problem and to focus on the topology of the problem space, graphs are often used to represent problem spaces. The states of the space are represented by nodes of the graph and the operators by edges between nodes. If an operator maps state  $x$  to state  $y$ , it will be represented by a directed edge from node  $x$  to node  $y$ . Edges may also be undirected, depending on whether their corresponding operators are invertible. If a single operator can be applied to a large number of different states, as in Rubik's Cube, that operator will be represented by a large number of different edges. This graph abstraction reduces problem solving to finding a path in a graph from the initial node to a goal node.

Although most problems are actually graphs with more than one path between a pair of nodes, for simplicity they are sometimes represented as trees where the initial state is the root of the tree. The cost of this simplification is that the same state may be represented by more than one node in the tree.

The two parameters of a search tree that determine the efficiency of various search algorithms are its branching factor and its depth. The branching factor ( $qv$ ) is the average number of descendants of a given node or the average number of new operators that are applicable to a given state, excluding the operator applied to generate the state. The depth is the length of the shortest path from the initial state to a goal state or the length of the shortest sequence of operators that solves the problem.

Three important special cases of problem spaces are considered: state spaces, problem-reduction spaces, and game trees.

**State Spaces.** The Rubik's Cube problem space mentioned above is an example of the simplest type of problem space, the state space. In a state space the nodes represent actual configurations of the problem to be solved, and the edges represent primitive actions in the problem domain. The solution to a state-space formulation of a problem is a simple path from the initial state to a goal state. A state-space graph is also called an OR graph because at any given node the problem solver need only apply one of the operators represented by the edges branching from that node.

Common examples in the AI literature of state-space problems are the Eight Puzzle and its larger relative the Fifteen Puzzle (see Fig. 1). The Eight Puzzle consists of a  $3 \times 3$  square frame containing 8 numbered square tiles and an empty position called the blank. The Fifteen Puzzle is  $4 \times 4$  and contains 15 tiles. The legal operators are to slide any tile that is horizontally or vertically adjacent to the blank into the blank position. The problem is to rearrange the tiles from some random initial configuration into a particular desired goal configuration. In a state-space formulation of these problems, the nodes correspond to the different permutations of the tiles, and the edges correspond to the primitive legal moves. A solution is a sequence of primitive moves that maps the initial state to the goal state.

**Problem-Reduction Spaces.** Another type of problem space is a problem-reduction space (see Problem reduction). In a problem-reduction space the nodes represent complete subproblems, as in a pair of initial and goal states. The root node of the tree represents the original problem to be solved, whereas the leaf or terminal nodes of the tree represent problems that can be solved by a single primitive action in the problem space. The edges represent problem-reduction operators, which decompose a given problem into a set of subproblems. If only one of the subproblems needs to be solved to solve the main problem, the node is called an OR node. If all the subproblems must be solved to solve the main problem, the node is called an AND node. A problem-reduction space containing both AND and OR nodes is called an AND/OR tree (see AND/OR graphs). A solu-

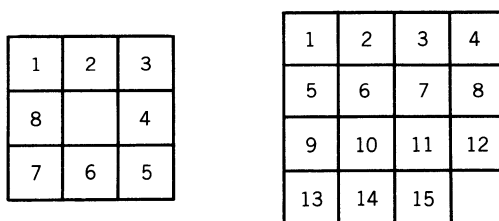


Figure 1. Eight and Fifteen puzzles.

tion to an AND/OR tree is a subtree that contains the root node, one branch from every OR node, and all the branches from every AND node. Due to space limitations, most of the algorithms in this entry are described only in terms of state spaces, but analogous algorithms exist for AND/OR graphs as well.

**Game Trees.** An important special case of an AND/OR tree is a two-player game tree. (see Game trees). In a game tree the nodes represent particular game situations, and the edges represent legal moves. The root node represents the initial game situation, and the leaves represent situations that are won, lost, or drawn positions. The edges descending from the root represent the legal moves of the first player to move, and thereafter alternate levels of the tree represent moves for one player or the other. The reason it is an AND/OR tree is that from the perspective of one of the players, his moves represent OR nodes since the choice of only one winning move guarantees a win. His opponent's moves represent AND nodes since he must consider all his opponent's responses and secure a win in each one (for game-tree searching algorithms, see Game playing, Game trees, Minimax procedure, and Alpha-beta pruning).

### Brute-Force Search

The most general search algorithms are brute-force searches since they do not require any domain-specific knowledge. All that is required for a brute-force search is a state description, the set of legal operators, the initial state, and a description of the goal state. The most important brute-force techniques are breadth-first, depth-first, depth-first iterative-deepening, and bidirectional searches.

**Breadth-First Search.** Breadth-first search begins by generating all the successors of the root node (this is known as expanding a node). Next, all the successor nodes are expanded, generating all the nodes at level 2 in the search tree. Breadth-first search continues by expanding one complete level of the tree at a time until a solution is found (see Fig. 2a).

Since breadth-first search never generates a node in the tree until all the nodes at shallower levels have been generated, once a path to a goal is found, it will be a path of shortest length. Thus, breadth-first search always finds an optimal solution by this measure.

The amount of time used by breadth-first search is a function of the branching factor  $b$  and the solution depth  $d$ . Since the amount of time is proportional to the number of nodes

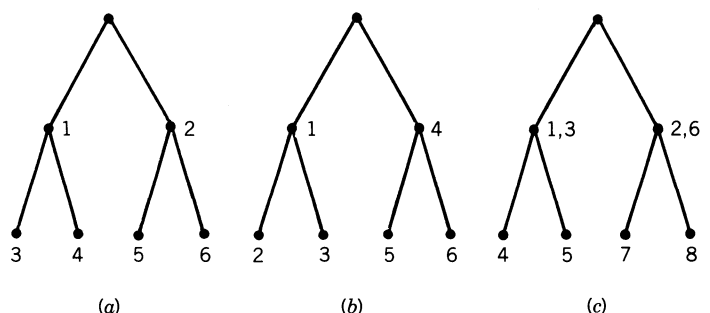


Figure 2. Order of node generation for (a) breadth-first, (b) depth-first, and (c) depth-first iterative-deepening searches.

generated, and the number of nodes at level  $d$  is  $b^d$ , the amount of time used by breadth-first search is

$$1 + b + b^2 + \dots + b^d$$

in the worst case. For large values of  $d$  this sum can be approximated by  $b^d$ , and hence the time complexity of breadth-first search is of order  $b^d$  or  $O(b^d)$ , which is an exponential function of  $d$ .

The main drawback of breadth-first search, however, is its memory requirement. Since each level of the tree must be saved in its entirety in order to generate the next level, and the amount of memory is proportional to the number of nodes stored, the space complexity of breadth-first search is also  $O(b^d)$ . As a result, breadth-first search is severely space-bound in practice and will exhaust the available memory in a matter of seconds on typical computer configurations.

**Depth-First Search.** An algorithm that remedies the space limitation of breadth-first search is depth-first search (see Search, depth-first). Depth-first search proceeds by first generating one successor of the root node, then generating one of its successors, and continuing to extend this single path until it terminates naturally or is cut off at some depth. Next, it backs up and generates another successor of the last node generated on the current path until that path is terminated (see Backtracking). In general, depth-first search expands nodes in a last-generated, first-expanded order (see Fig. 2b).

Since a depth-first search to depth  $d$  generates the same set of nodes as a breadth-first search, albeit in a different order, the time complexity of depth-first search is also  $O(b^d)$ . Its advantage, however, lies in its space efficiency. Since depth-first search only needs to store the current path in order to continue, and the maximum length of this path is only  $d$  nodes, the space complexity of depth-first search is  $O(d)$ , which is a linear function of  $d$ . As a practical matter, depth-first search is time-limited rather than space-limited.

However, the disadvantage of depth-first search is that in general it requires an arbitrary cutoff depth in order to terminate. Without such a cutoff depth the first path to be explored could go on forever, eventually cycling back to revisit previous states in an infinite loop. Although the ideal cutoff is the solution depth  $d$ , this value is almost never known in advance of actually solving the problem. If the chosen cutoff depth  $c$  is less than the solution depth  $d$ , the algorithm will terminate without finding a solution, whereas if  $c$  is greater than  $d$ , a large price,  $O(b^c)$ , is paid in terms of execution time, and the first solution found may not be an optimal one.

**Depth-First Iterative-Deepening.** An algorithm that rectifies the drawbacks of both breadth-first and depth-first search is called depth-first iterative-deepening. Depth-first iterative-deepening (DFID) first performs a depth-first search to depth 1, then a complete depth-first search to depth 2, and continues executing depth-first searches, increasing the depth cutoff by 1 at each iteration, until a solution is found (see Fig. 2c).

Since it never generates a node until all shallower nodes have been generated, the first solution found by DFID is guaranteed to be an optimal one. Furthermore, since at any given point it is executing a depth-first search, the space complexity of DFID is only  $O(d)$ . Finally, although it appears that DFID wastes a great deal of time in the iterations prior to the one that finds a solution, it can be shown that the time complexity of DFID is in the same class as that of breadth-first search,

namely  $O(b^d)$ . The intuitive reason for this is that since the number of nodes in a given level of the tree grows exponentially with depth, almost all the time is spent in the deepest level, even though shallower levels are generated an arithmetically increasing number of times.

It can be proven by a simple adversary argument that any brute-force search algorithm must take  $O(b^d)$  time. Furthermore, a simple information-theoretic argument shows that any algorithm that takes  $O(b^d)$  time must use  $O(d)$  space. These facts imply that DFID is asymptotically optimal in terms of time and space among all brute-force search algorithms that find optimal solutions (2).

**Bidirectional Search.** One final brute-force search algorithm, which requires some additional structure in the problem space, is called bidirectional search (3) (see Search, bidirectional). The main idea of bidirectional search is that instead of blindly searching forward from the initial state until the goal is reached, or even blindly searching backward from the goal until one encounters the initial state, one could search from both initial and goal states until the two searches meet in the middle. Bidirectional search executes two breadth-first searches, one from the initial state and one from the goal state, until a common state is found among the two search frontiers. Then the path from the initial state is concatenated with the inverse of the path from the goal state to form the complete solution path.

Bidirectional search requires an explicit goal state rather than a goal criterion, and that the operators of the problem space be invertible, or that backward chaining (see Processing, bottom-up and top-down) can be done. Assuming that the comparisons for identifying common states can be done in constant time per node, by using a technique such as hashing, the time complexity of bidirectional search is  $O(b^{d/2})$  since each search need only proceed to half the solution depth. Since at least one of the searches must be breadth-first in order to find a common state, the space complexity of bidirectional search is also  $O(b^{d/2})$ .

### Heuristic Search

All brute-force algorithms suffer in efficiency from the fact that they are essentially blind searches; they use no domain knowledge to guide the choice of which nodes to expand next. The idea of heuristic search is based on the fact that most problem spaces provide information, at a small computational cost, that distinguishes among states in terms of their likelihood of being close to a goal. This information is called a heuristic (qv). For example, if one were navigating in a street network toward a goal, in the absence of further information one would give preference to streets that proceed in the direction of the goal state, bearing in mind that this approach is not foolproof due to the possibility of intervening obstacles such as rivers or mountains.

A heuristic evaluation function is a special case of a heuristic that is a function from a pair of states to a number that estimates the distance in the problem space between the two states. For example, a common heuristic function for the Eight Puzzle introduced earlier is called Manhattan distance. It is computed by counting, for each tile not in its goal position, the number of moves along the grid it is away from its goal position, and summing these values over all tiles. The Manhattan-distance heuristic is an estimate of the number of moves re-

quired to get from one state to another and can be computed much faster than by actually solving the problem and counting the moves. The navigation heuristic of going in the right direction can be captured by a heuristic evaluation function that measures the airline distance between a pair of states, and minimizes this value.

A number of different algorithms make use of heuristic evaluation functions (see Search, branch-and-bound), including hill-climbing, best-first search,  $A^*$ , beam search, and iterative-deepening- $A^*$ . Each of these is examined in turn, starting with the simplest and proceeding to the more complex. In addition, heuristic information can be employed in bidirectional search as well (for a description of this technique, see Search, bidirectional).

**Hill-Climbing.** The simplest heuristic search algorithm is called hill-climbing. The term comes from the analogy of a problem space to a two-dimensional plane where the heuristic function defines the elevation of each point and the goal is to reach the highest point on the terrain. Hill-climbing works by always moving in the direction of locally steepest ascent. The only memory is a current state that is originally the initial state. At each cycle the current state is expanded, the heuristic function is applied to each of its successors, and the one with the best heuristic value becomes the new current state, replacing the previous one.

There are a number of problems with this algorithm, the most serious of which is that it may never terminate. Since the algorithm only stores a single state, it has no way of knowing where it has been and thus runs the risk of returning to a previously visited state and looping forever. For example, if the hill-climber encountered a volcanic cone, it would climb to the rim of the crater and then go around the rim indefinitely.

This problem can be rectified by modifying hill climbing as follows: Maintain a list, typically called a CLOSED list, of states that have been visited and never revisit a closed state. This guarantees that the algorithm will terminate in any finite problem space.

Unfortunately, this modified hill-climbing algorithm still suffers from the classic foothill problem: A local maximum is not always the global maximum. This difficulty is remedied by an algorithm called best-first search (see Search, best-first).

**Best-First Search.** In addition to a CLOSED list, best-first search also maintains an OPEN list of states that have been generated but not yet expanded. This gives the algorithm the ability to resume a path that was temporarily abandoned in favor of a path that appeared more promising, if the new path fails to live up to its expectations. It works as follows: The best node on the OPEN list, initially just the start state, is expanded, generating all of its successors, and then is placed on the CLOSED list. The heuristic evaluation function is applied to all the successors and they are inserted into the OPEN list in order of their heuristic values. In the next cycle the best node on the OPEN list is chosen for expansion.

Assuming that low values of the heuristic function are best, as indeed they would be if the function measured distance to the goal, the behavior of best-first search is analogous to water flowing over a surface seeking the lowest point. The water originates from a spring at the initial state and follows the path of locally steepest descent. When it enters a local minimum, it forms a lake that rises until its level reaches that of an outlet. The CLOSED list corresponds to the surface under-

water, and the OPEN list corresponds to the shoreline. The search always proceeds from the lowest point on the shoreline.

Like water flowing over a surface, best-first search is guaranteed to eventually find the global minimum. Even if it occurs in the crater of a high volcano, once the water reaches the level of the rim of the crater, it will flow into the lowest point. Unfortunately, however, in the same way that a river is rarely the shortest path between two points, best-first search is not guaranteed to find a shortest path to the goal. The reason for this is that selection of a node to expand next is based solely on the estimate of its distance to the goal and does not take into account its distance from the initial state. Thus, given two states, one of which is a long way from the initial state but has a slightly shorter estimate of distance to the goal and another that is very close to the initial state but has a slightly longer estimate of distance to the goal, best-first search will always choose to expand next the state with the shorter estimate. The famous  $A^*$  algorithm (qv) is an attempt to rectify this situation.

$A^*$ .  $A^*$  is a best-first search algorithm in which the figure of merit associated with a node is not just the heuristic estimate but rather  $f(n) = g(n) + h(n)$ , where  $g(n)$  is the cost of the path from the initial state to the node  $n$ , and  $h(n)$  is the heuristic estimate of the cost of the cheapest path from the node  $n$  to a goal node. In other words,  $f(n)$  is an estimate of the total cost of the cheapest solution path going through node  $n$ . At each point the node with the lowest  $f$  value is chosen for expansion next.

An interesting and somewhat surprising theorem is that  $A^*$  will always find an optimal path to a goal if the heuristic function  $h(n)$  never overestimates the actual distance to the goal (4). For example, it should be clear that the Manhattan-distance heuristic never overestimates the distance to the goal in the Eight Puzzle. Therefore, the first solution found by  $A^*$  using Manhattan distance for  $h(n)$  will be an optimal one. Similarly, airline distance never overestimates actual highway distance, so  $A^*$  with the airline-distance heuristic will find optimal solutions to the road-navigation problem.

This result is easiest to see if the cost function  $f(n)$  is monotonic, or never decreases along a path away from the initial state. Monotonicity is equivalent to the heuristic function  $h(n)$  being consistent, or obeying the triangle inequality of metrics. In either case  $A^*$  expands paths in nondecreasing order of cost, and hence, once it expands a goal node, it will have found a lowest cost path to a goal.

A less widely known property of  $A^*$  is that it makes the most efficient use of a given heuristic function in the following sense: Among all algorithms using a given consistent heuristic function  $h(n)$  and that find an optimal solution,  $A^*$  expands the fewest number of nodes, up to tie-breaking among nodes of equal cost (5). The actual number of nodes expanded by  $A^*$  depends on the accuracy of the heuristic function. For example, if the heuristic function exhibits constant absolute error, the number of nodes expanded is a linear function of the solution depth (6), whereas if the heuristic is subject to constant relative error, a much more realistic assumption, the number of node expansions is exponential in the solution depth (7).

The main drawback of  $A^*$ , and indeed of any best-first search, is its memory requirement. Since the entire OPEN list must be saved,  $A^*$  is severely space-limited in practice and is no more practical than breadth-first search on current machines. There are two solutions to this problem: beam search and iterative-deepening- $A^*$  (8).

**Beam Search.** Beam search (see Beam search) reduces the memory load of  $A^*$  by only retaining the best  $n$  nodes of the OPEN list for some value of  $n$ . When the OPEN list is full and a new node must be stored, the worst node is dropped off the list and removed from further consideration. The cost of this technique is the loss of solution optimality: The only node on OPEN that leads to an optimal solution could be pruned in this fashion, and then any solution found would be suboptimal.

**Iterative-Deepening- $A^*$ .** Alternatively, in the same way that depth-first iterative-deepening defeated the space complexity of breadth-first search, iterative-deepening- $A^*$  (IDA\*) drastically reduces the memory requirement of  $A^*$  without sacrificing optimality of the solutions found (2). Each iteration of the algorithm is a complete depth-first search that keeps track of the cost,  $f(n) = g(n) + h(n)$ , of each node generated. As soon as this cost exceeds some threshold, that branch is cut off and the search backtracks to the most recently generated node. The cost threshold starts with the heuristic estimate of the initial state and in each successive iteration is increased to the minimum value that exceeded the previous threshold.

Since at any point IDA\* is performing a depth-first search, the memory requirement of the algorithm is linear in the solution depth. In addition, it can easily be shown that if the cost function is monotonic, the first solution found by IDA\* is an optimal one. This condition is not really a restriction since given a nonmonotonic cost function, a monotonic one can trivially be constructed by taking the maximum value of the function along the path from the initial state to a given node to be the value of that node. Finally, by an argument similar to that presented for depth-first iterative-deepening, it can be shown that IDA\* expands the same number of nodes, asymptotically, as  $A^*$  provided that the number of nodes grows exponentially with solution depth and that the number of duplicate nodes in the search tree is relatively small. In practice, both of these assumptions are quite realistic. These facts, together with the optimality of  $A^*$ , show that in practice IDA\* is asymptotically optimal in terms of time and space over all heuristic search algorithms that find optimal solutions.

An additional benefit of IDA\* is that it is easier to implement and runs faster per node than  $A^*$ . This is because it is a simple depth-first search and does not incur the overhead of managing an OPEN list. In experiments IDA\* has proven to be the only algorithm capable of finding optimal solutions to the Fifteen Puzzle within practical time and space limits.

## BIBLIOGRAPHY

1. A. Newell and H. A. Simon, *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ, 1972.
2. R. E. Korf, "Depth-first iterative-deepening: An optimal admissible tree search," *Artif. Intell.* **27**(1), 97-109 (1985).
3. I. Pohl, Bi-Directional Search, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 6, American Elsevier, New York, pp. 127-140, 1971.
4. P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybernet.* **4**(2), 100-107 (1968).
5. R. Dechter and J. Pearl, "Generalized best-first search strategies and the optimality of  $A^*$ ," *J. Assoc. Comput. Mach.* **32**(3), 505-536 (July 1985).
6. I. Pohl, First results on the Effect of Error in Heuristic Search, in B.

Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 5, American Elsevier, New York, pp. 219-236, 1970.

7. J. Gaschnig, Performance Measurement and Analysis of Certain Search Algorithms, Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, May 1979.
8. J. Pearl, *Heuristics*, Addison-Wesley, Reading, MA, 1984. This is the most complete reference available on search algorithms.

R. E. KORF  
UCLA

This work was supported in part by NSF Grant IST 85-15302, by an NSF Presidential Young Investigator Award, and by an IBM Faculty Development Award.

**SEARCH BEAM.** See Beam search.

## SEARCH, BEST-FIRST

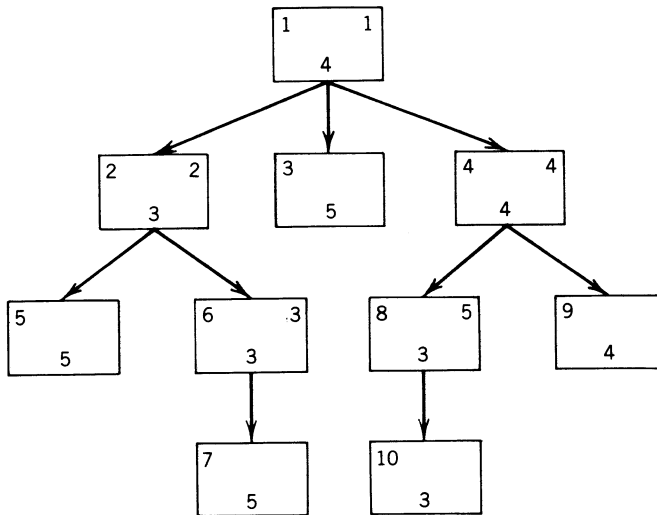
Many times, the task of solving a problem—i.e., of achieving some desired situation—can be formulated as a search (qv) in a directed graph of arcs and nodes for a path from an initial node to a desired node. The initial node represents the current situation; the final, or goal, node represents the desired situation; and the remaining nodes on the path represent intermediate situations. Each directed arc from one node to another represents an operation that changes the state of the world from that represented by the first node to that represented by the second. In a forward search, processing begins with the generation of the initial node's successors—the nodes one step away via the directed arcs. On each subsequent search step a new node from the graph is selected and its successors are generated. The search terminates when the goal node is found. Best-first search, sometimes called ordered search, is a way to minimize the number of nodes processed during the search by bringing knowledge to bear on the node-selection process. The choice is determined by an evaluation function that characterizes the relative "goodness" of the nodes. On each cycle of the search the node selected is the best one that has been generated but not yet processed.

Figure 1 shows a tree-structured graph as it would be traversed by a best-first search. During the first step of the search the initial node's successors (nodes 2, 3, and 4) are generated. On the second cycle the best unprocessed node (node 2) is selected, generating its successors (nodes 5 and 6). On the third cycle the evaluations of all of the unprocessed nodes (nodes 3, 4, 5 and 6) are compared, and the best one (node 6) is selected, generating node 7. Up until this point the node selected has always been a successor of the previous node; however, the next node selected (node 4) is in a completely different region of the tree. Unlike hill climbing or depth-first search, best-first search does not limit the selection to the successors of the currently selected node; instead it picks the best unprocessed node in the tree. On the next cycle one of the successors of node 4 (node 8) is selected, generating the goal node (node 10).

## Algorithm

Figure 2 contains a best-first search algorithm under the simplifying assumption that the actual path from the initial node to the goal node is not needed; all that is required is that the





**Figure 1.** A best-first search of a tree-structured graph. Three numbers are associated with each node: the order in which the node is generated (top left corner of node); the order in which the node is processed (top right corner of node); and the evaluation of the node (lower is better) (bottom of node). The first node generated is the initial node and the last node generated (node 10) is the goal node.

```

Node ← Initial-Node
While Not Goal(Node) Do
  Begin
    For-Each Successor of Node Do
      Insert-Into-Priority-Queue(Successor, Evaluate(Successor))
    Node ← Remove-Best-From-Priority-Queue()
  End

```

**Figure 2.** A simple best-first search algorithm.

goal node be generated. Current best-first search algorithms are based on the algorithm originally published as part of the work on the Graph Traverser program (1). The primary data structure in this algorithm is a priority queue (see Ref. 2 for efficient implementations of priority queues). The algorithm's main loop consists of selecting the best node in the queue, checking if it is a goal node, generating the node's successors, and inserting the successors into the queue according to their evaluations.

This simple algorithm reveals the essential structure of the search but is inadequate for most real cases. A number of more complex variations exist that increase either the search's efficiency or its functionality. (For good discussions of best-first search and examples of algorithms based on these variations, see Refs. 3 and 4.) Here are four of the most important variations.

1. If the goal node is not guaranteed to have the best evaluation, unnecessary nodes will be processed in between when the goal node is generated and when it is selected. This can be avoided by comparing each node with the goal as soon as it is generated.
2. If the path from the initial node to the goal node is required, at each step of the search backward pointers to the node can be created from the node's successors. Once a goal

node is reached, the backward pointers can be traversed to find the path.

3. If the search space is a general directed graph (as opposed to a tree), the algorithm will reprocess a node each time it is reached in the search. This can be avoided by keeping a list of all of the processed nodes, and only inserting a new node into the queue if it has not yet been processed and is not already in the queue.
4. If the evaluation of a node depends on the path by which it is reached, not all instances of a node are equivalent, and not all paths to the node are equivalent.

Under the circumstances in variation 4 the just described procedure for handling general directed graphs must be replaced by the following more complex procedure.

If the node has not been seen before, insert it into the priority queue.

If the node has been seen before and the new evaluation is worse than the old one, throw away the new instance.

If there is an old instance of the node already in the queue and the new evaluation is better than the old one, delete the old instance from the queue, insert the new instance into the queue, and delete the backward pointer to the old instance's predecessor.

If the node has already been processed and the new evaluation is better than the old one, follow one of the following two strategies: Reprocess the node; that is, insert the new instance into the queue, and delete the backward pointer to the old instance's predecessor; or update the node and its successors (this somewhat complicated procedure is described in Ref. 4).

## Related Search Strategies

Hill climbing: Restrict the nodes eligible for selection to the successors of the currently selected node.

Beam search (qv): The size of the priority queue is bounded.

Breadth-first search: The evaluation of a node is the distance from it to the initial node. The smaller the distance, the better.

Depth-first search (qv): The evaluation of a node is the distance from it to the initial node. The greater the distance, the better.

A\* (qv): The evaluation function takes the form  $f(n) = g(n) + h(n)$ , where  $g(n)$  is the cost of reaching node  $n$  from the initial node, and  $h(n)$  is an estimate of the cost of reaching the goal node from node  $n$  (5).

AO\*: A variant of the A\* algorithm for AND/OR graphs (qv).

## BIBLIOGRAPHY

1. J. E. Doran and D. Michie, Experiments with the Graph Traverser program, *Proc. Roy. Soc. Lond.* **294(A)**, 235-259 (1966).
2. D. E. Knuth, *The Art of Computer Programming*, Vol. 3, *Sorting and Searching*. Addison-Wesley, Reading, MA, 1973.
3. J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, Reading, MA, 1984. Contains a

very detailed and thorough explication of best-first search and related algorithms.

4. E. Rich, *Artificial Intelligence*, McGraw-Hill, New York, 1983.
5. P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Sys. Sci. Cybernet.* 4, 100-107 (1968).

P. ROSENBLOOM  
Stanford University

## SEARCH, BIDIRECTIONAL

Bidirectional search is a generalization of forward-search algorithms in domains where an explicit goal state is known. Ordinary heuristic-search procedures such as A\* (see A\* algorithm) are forward searches (1,2) from a start node to a goal node. Bidirectional search is a technique combining both forward and backward search (3).

Bidirectional search as applied to AI problems was first studied in the late 1960s (4). It was used to compute 15 puzzle solutions on a class of benchmark problems which were an outgrowth of the graph traverser research at the University of Edinburgh (5).

### Strategies in Bidirectional Search

In the ordinary shortest path problem, bidirectional search can be shown to be more efficient than comparable unidirectional methods. The search out of each direction can be thought of as a spherical wave front. It has been shown that expanding in the "sparser" direction (fewest open nodes) is an optimally efficient search in the absence of other information.

In the nonheuristic shortest path problem bidirectional search achieves significant computational savings. The motivation is that search trees grow exponentially as a function of the average branching factor of the space, and therefore a search that explores two search trees of shorter depth generates far fewer nodes than a single search tree of longer depth.

In heuristic search a selective portion of the available space is explored. Early experiments found bidirectional heuristic search to be frequently twice the effort of ordinary heuristic search (6). The selectivity that makes heuristic search efficient defeats the additional computational savings expected from bidirectional search.

Several methods have been proposed to remedy this defect. One attack is the bidirectional heuristic front-to-front algorithm (BHFFA) (7). This algorithm considers all pairs of nodes  $(x, y)$  with  $x$  in the forward open set and  $y$  in the backward open set. The algorithm chooses that node pair that minimizes

$$f(x, y) = g(x) + h(x, y) + g(y)$$

where  $g(x)$  is the actual distant from the start node to the node  $x$ ,  $g(y)$  is the actual distant from the node  $y$  to goal node, and  $h(x, y)$  is the heuristically estimated distance from  $x$  to  $y$ . This produces a solution that expands fewer nodes than other search schemes at the considerable expense of computing many evaluation functions per node pair in the cross product.

### Retargeting

A computationally efficient scheme is the d-node retargeting method (6). In this algorithm the search is conducted in one

direction for a fixed number of iterations. At this point the algorithm reverses direction. However, rather than aiming its search at the start node, it targets a d-node. This node is the deepest leaf node. Each time the direction of search is changed, a new d-node is computed to be the target for the opposite search.

Each direction employs a standard unidirectional search algorithm. The number of iterations before reversing is crucial to the efficiency of this method. It must be large enough to guarantee that a suitable d-node is selected and small enough to take advantage of the improved target node. Practice has shown that the retargeting method is the computationally most efficient available bidirectional scheme (see also Processing, bottom-up and top-down).

## BIBLIOGRAPHY

1. P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybernet.* SSC-4(2), 100-107 (1968).
2. I. Pohl, "Heuristic search viewed as a path problem," *Artif. Intell.* 1, 193-204 (1970).
3. I. Pohl, Bi-Directional Search, in B. Meltzer and D. Michie (eds.), *Machine Intelligence* Vol. 6, American Elsevier, New York, pp. 219-236, 1971.
4. J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, Reading, MA, 1984.
5. J. Doran and D. Michie, "Experiments with the graph traverser program," *Proc. Roy. Soc. Lond.* 294(A), 235-259 (1966).
6. G. Politowski and I. Pohl, D-Node Retargeting in Bidirectional Heuristic Search, *Proceedings of the Fourth AAAI Conference*, Austin, TX, pp. 274-277, 1984.
7. D. De Champeaux, "Bidirectional heuristic search again," *JACM* 30(1), 22-32 (January 1983).

I. POHL  
University of California at Santa Cruz

## SEARCH, BRANCH-AND-BOUND

Branch and bound (B&B) is a problem-solving technique that has been usefully employed in various problems encountered in operations research and combinatorial mathematics. Many heuristic search (qv) procedures used in AI can be viewed as B&B procedures. The class of problems solved by B&B procedures can be abstractly stated as:

Given a (possibly infinite) discrete set  $X$  and a real-valued cost function  $f$  whose domain is  $X$ , find an optimal element  $x^*$  of  $X$  such that  $f(x^*) = \min\{f(x) | x \in X\}$ . Discussion in this entry is also applicable (with appropriate modifications) to the case when a largest cost element is desired; i.e., an optimal element is defined as an element  $x^*$  of  $X$  such that  $f(x^*) = \max\{f(x) | x \in X\}$ .

Unless there is enough problem-specific knowledge available to obtain an optimum element of the set in some straightforward manner, the only available alternative may be to implicitly enumerate the set  $X$ . For practical problems, the size of the set  $X$  is quite large, which makes exhaustive enumeration prohibitively time-consuming.

Using the available knowledge about the problem, B&B procedures decompose the original set into sets of smaller and smaller sizes. The decomposition of each generated set  $S$  is continued until tests reveal either that  $S$  is a singleton (in which case its value is measured directly and compared with the currently best member's cost) or that there is an optimum element  $x^*$  not in  $S$  (in which case the set is "pruned" or eliminated from further consideration). If the decomposition process is continued (and satisfies some properties), an optimum element will eventually be found. The utility of this approach derives from the fact that, in general, most of  $X$  will be pruned, whence only a small fraction of  $X$  need be enumerated. B&B is commonly used to solve such important problems as integer programming, scheduling problems, network problems, the traveling-salesman problem, the knapsack problem, and the shortest path problem.

B&B techniques appear to have been conceptualized in the early sixties to tackle integer programming and nonlinear-assignment problems. Later similar techniques, with some modifications, were found to be applicable in many other problem domains. As more and more applications were discovered, the B&B methodology evolved. Various formal models of B&B were presented and later superseded by more general models (1-6). In the earlier formulations of B&B (1,2) only the lower and upper bounds on the costs of the elements of (sub)sets (of  $X$ ) were used for pruning. If two sets  $X_1$  and  $X_2$  are in the collection of sets under consideration, and the lower bound on the costs of elements in  $X_1$  is no smaller than the upper bound on the costs of elements in  $X_2$ ,  $X_1$  can be pruned. The use of bounds for pruning gave the procedure its name branch and bound. The concept of pruning by bounds was later generalized to include pruning by "dominance" (see Refs. 3-6). Dominance may be used to perform pruning even when adequate lower bound information is not available. Pruning by dominance provides a powerful mechanism for using problem-specific knowledge for efficient search of an optimum element of the set  $X$ .

### A General B&B Formulation

Here the basic elements of a B&B formulation are briefly described. The formulation presented here is very similar to the one given in Ref. 7. As is discussed in Ref. 7, the dominance relation in the formulation is used for pruning in a manner somewhat different than in Refs. 3-5.

**Basic Definitions.** Let  $Y$  be the set of all subsets of  $X$ , i.e.,  $Y = 2^X$ .  $X_i$  denotes a subset of  $X$ , and  $A$  denotes a collection of subsets of  $X$  (i.e.,  $A \subseteq Y$ ). For brevity,  $A$  is sometimes referred to simply as a "collection." For notational convenience, the union of all subsets in any collection  $A$  is denoted by  $\cup(A)$ ; i.e.,  $\cup(A) = \cup\{X_i | X_i \in A\}$ . The quantity  $f^*(X_i)$  is defined to be the minimum of the costs of the elements in  $X_i$ . Any element  $x^* \in X_i$  such that  $f(x^*) = f^*(X_i)$  is called an optimum element of  $X_i$ .

A branching function **BRANCH** is any function that divides the members of the collection  $A$  into subsets that collectively include precisely the same elements of  $X$  as the original collection  $A$ . Mathematically, it is any function mapping collections into collections such that:

1.  $X_i \in \text{BRANCH}(A) \Rightarrow X_i \subseteq X_j$  for some  $X_j \in A$ , and
2.  $\cup(\text{BRANCH}(A)) = \cup(A)$ .

Often the function **BRANCH** is defined as a composition of selection and splitting functions. A selection function is any function **SELECT** mapping collections into collections such that  $\text{SELECT}(A) \subseteq A$ . A splitting function **SPLIT** is any function satisfying the properties of a branching function. **BRANCH** is then defined as

$$\text{BRANCH}(A) = (A - \{\text{SELECT}(A)\}) \cup \text{SPLIT}(\text{SELECT}(A))$$

Although this definition of **BRANCH** is mathematically equivalent to the one given above, it emphasizes the characteristic that only the elements from a certain selected subset of the collection  $A$  are divided, and the rest are returned unchanged. In fact, in many implementations of **BRANCH**, only one selected element from the collection  $A$  is divided, and the rest are returned unchanged.

The dominance relation  $D$  is the binary relation between subsets  $X_i, X_j$  of  $X$  such that  $X_i D X_j$  iff  $f^*(X_i) \leq f^*(X_j)$ . Clearly, if  $X_i$  and  $X_j$  are present in a collection  $A$  and  $X_i$  dominates  $X_j$ , then  $X_j$  can be pruned.

The pruning function **PRUNE** prunes a dominated subset of  $A$ . It is defined as  $\text{PRUNE}(A) = A - A^D$ , where  $A^D$  is a subset of  $A$  such that for all  $X_i \in A^D$  there exists some  $X_j \in A - A^D$  such that  $X_j D X_i$ . From the definition of dominance, it follows that  $A - A^D$  will contain at least one optimal element of  $\cup(A)$ .

**An Abstract B&B Procedure.** The procedure **BB** given below represents the essence of many B&B procedures. Here,  $A$  denotes the collection of subsets of  $X$  upon which the branching and pruning operations are performed in each iteration of **BB**, and  $|S|$  denotes the cardinality of a set  $S$ .

procedure **BB** (\* B&B procedure to search for an optimum element of a set  $X$  \*)

begin

$A := \{X\}$ ; (\* initialize the collection  $A$  \*)

while  $|\cup(A)| \neq 1$  do (\* loop until  $A$  contains only one element of  $X$  \*)

$A := \text{BRANCH}(A)$ ; (\* branch on the collection  $A$  \*)

$A := \text{PRUNE}(A)$  (\* eliminate the dominated subsets

end

end

No element of  $X$  is lost from  $A$  in the branching operation, and at least one optimal element of  $\cup(A)$  is there in  $\text{PRUNE}(A)$ . Hence, if the procedure **BB** terminates,  $A$  contains only an optimal element of  $X$ . Note that the termination of **BB** is not guaranteed. In order to guarantee the termination of **BB**, **BRANCH**, and **PRUNE** must satisfy certain additional properties.

**Best-First B&B.** In many problem domains it is possible to define a function **lb** on the subsets  $X_i$  of  $X$  such that

1. for all  $x \in X_i$ ,  $\text{lb}(X_i) \leq f(x)$ ; i.e.,  $\text{lb}(X_i)$  is a lower bound on the costs of the elements of  $X_i$ ; and
2. for all  $x \in X$ ,  $\text{lb}(\{x\}) = f(x)$ ; i.e., the lower bounds for singleton sets are not unnecessarily loose.

This lower bound information can be fruitfully used in selecting an element for branching. If in every cycle of BB's loop an element of  $A$  is chosen for branching that has the least lower bound of all the elements of  $A$ , the selection rule is called best-first, and the B&B procedure using this strategy is called best-first B&B (see Search, best-first). An interesting feature of best-first B&B is that whenever a singleton set  $\{x\}$  is selected for branching, the procedure can terminate. This is because  $f^*(\{x\}) = f(x) = \text{lb}(\{x\}) \leq \text{lb}(X_i) \leq f^*(X_i)$  for all  $X_i \in A$ , and thus  $\{x\}$  dominates all the other elements in  $A$ . If more than one element of  $A$  is selected for branching, the selection rule is still called best-first as long as at least one of the selected elements (e.g.,  $X_1$ ) has the least lower bound of all the elements of  $A$ . In such cases B&B can successfully terminate if  $X_1$  is a singleton.

If the bounds  $\text{lb}(X_i)$  are good approximations of  $f^*(X_i)$ , best-first B&B can be very efficient. In the extreme case, if  $\text{lb}(X_i) = f^*(X_i)$  for all  $X_i \subseteq X$ , the B&B procedure finds an optimal element of  $X$  by splitting only those sets that contain optimal elements. On the other hand, if the bounds  $\text{lb}(X_i)$  are not good approximations of  $f^*(X_i)$ , best-first B&B can be very inefficient and may require a lot of storage.

**Depth-First B&B.** Another way of selecting a set for branching from the active collection  $A$  is to select a set from those sets that have been generated most recently as a result of branching (8). This selection rule is called depth-first, and the B&B procedure using such a rule is called depth-first B&B (see Search, depth-first). The major advantage of the depth-first selection rule over the best-first selection rule is that, in general, it requires less storage. But the depth-first search, being a uninformed search, can be much slower than the best-first search. See Ref. 9 for a theoretical comparison of various search strategies used in B&B procedures.

**More on Branching and Pruning.** In this abstract formulation a number of details have been left out. For example, only the basic properties of a branching function have been defined. In a practical implementation of a B&B procedure, a branching function is chosen that is natural for the problem domain in question and satisfies the properties given here. The splitting function is usually suggested by the problem. But sometimes, a problem may have many natural splitting functions [e.g., the traveling-salesman problem has many natural splitting functions (10)]. The selection function can be depth-first, best-first, or some other function suitable for the problem at hand.

For pruning, in each cycle of BB, a dominated subset  $A^D$  of the collection  $A$  needs to be constructed. Note that for any two subsets  $X_1, X_2$  of  $X$ , at least one of them dominates the other [either  $f^*(X_1) \geq f^*(X_2)$  or  $f^*(X_2) \geq f^*(X_1)$ ]. Hence, in theory,  $A^D$  could be constructed to have all but one set of the collection  $A$ . This would make the procedure BB terminate in a very few cycles since in every cycle of BB all but one of the generated sets is eliminated. In practice, it may not be known which sets in  $A$  dominate which other sets in  $A$  without exhaustively enumerating the elements in the sets that are members of  $A$ . However, partial knowledge from the problem domain is often available to reveal that certain sets in  $A$  dominate certain other sets in  $A$ . This partial knowledge of the dominance relation can be used to construct a dominated subset  $A^D$  of  $A$ .

Next a practical B&B procedure is presented that finds a shortest path in a directed graph. This also illustrates how

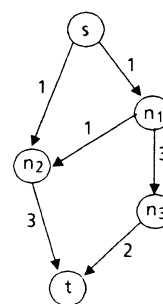


Figure 1. Steps of a best-first B&B search procedure using bounds for pruning.

domain knowledge about directed graphs can be used to devise practical branching and pruning functions. See Refs. 8, 10, and 11 for other applications of B&B.

### Examples

**Finding a Shortest Path in a Graph.** A directed graph  $G$  is given. Each arc  $(n, m)$  in  $G$  has a cost  $c(n, m) \geq 0$ , and for every path  $P$  in  $G$ ,  $\text{cost}(P)$  is defined as the sum of the arc costs of  $P$ . The problem is to find a least-cost directed path from a source node  $s$  in  $G$  to a terminal node  $t$  in  $G$ . For this problem  $X$  is the set containing each path from  $s$  to  $t$ . For  $x \in X$ ,  $f(x) = \text{cost}(x)$ .

If  $(m, n)$  is an arc in  $G$ ,  $n$  is called a successor of  $m$ . Suppose  $P = (n_1, n_2, \dots, n_j)$  is a path in  $G$ , then  $Pn$  is the path  $(n_1, n_2, \dots, n_j, n)$ . A path  $P$  from  $s$  to a node  $n$  is used to represent the set of paths in  $X$  that are extensions of  $P$ . [In actual implementations of B&B, the set  $X$  and its subsets are not represented explicitly. Instead, some problem-specific data structure is used that implicitly represents  $X$  and its subsets (12).] Let  $n_1, \dots, n_k$  be successors of  $n$  in  $G$ ; then a natural splitting function on  $P$  is  $\text{SPLIT}(P) = \{Pn_i \mid 1 \leq i \leq k\}$ . For a path  $P$  from  $s$  to a node  $n$ , a lower bound function  $\text{lb}$  is defined as  $\text{lb}(P) = \text{cost}(P)$ .

**Pruning Based on Lower Bounds.** Let  $P'$  be a path from  $s$  to  $t$ , and  $\text{cost}(P') \leq \text{lb}(P'')$ , where  $P''$  is some path between  $s$  and a node  $n$ . Clearly,  $P'$  is no worse than any path represented by  $P''$ ; i.e.,  $P'$  dominates  $P''$ . Hence if  $P'$  and  $P''$  are present in the collection  $A$  in BB,  $P''$  can be pruned. Figure 1 shows the operation of a best-first B&B procedure using this kind of pruning on the graph of Figure 2.

Figure 2a shows a path  $(s)$  that represents  $X$ , the set of all paths between  $s$  and  $t$ . At the beginning of BB,  $A$  is initialized to  $s$ . Figure 2b shows  $A$  after a branching operation is performed on  $s$ , which results in paths  $(s, n_1)$  and  $(s, n_2)$ . No pruning is performed in this cycle of BB. In the next cycle of BB the splitting operation is performed on  $(s, n_1)$  [because  $\text{lb}((s, n_1)) \leq \text{lb}((s, n_2))$ ], which results in  $(s, n_1, n_2)$  and  $(s, n_1, n_3)$ . Figure 2c shows  $A = \{(s, n_2), (s, n_1, n_2), (s, n_1, n_3)\}$  at this point. In the next cycle of BB,  $(s, n_2)$  is selected for splitting. The resulting path  $(s, n_2, t)$  dominates  $(s, n_1, n_3)$  because  $f(s, n_2, t) \leq \text{lb}(s, n_1, n_3)$ . Therefore,  $(s, n_1, n_3)$  is pruned (see Fig. 2d). In the next cycle of BB,  $(s, n_1, n_2)$  is selected for splitting, which results in  $(s, n_1, n_2, t)$ . As shown in Figure 2e,  $(s, n_2, t)$  dominates  $(s, n_1, n_2, t)$ ; hence  $(s, n_1, n_2, t)$  is pruned. Now  $A$  contains only  $(s, n_2, t)$ , which is a singleton set. Therefore, BB terminates with  $(s, n_2, t)$  as a shortest path between  $s$  and  $t$ .

**Pruning Based on Dominance.** If  $P'$  and  $P''$  are two paths between  $s$  and a node  $n$ , and  $\text{cost}(P') \leq \text{cost}(P'')$ ,  $P' \text{ DP}''$  because for any extension of  $P''$  there exists an extension of  $P'$  that has

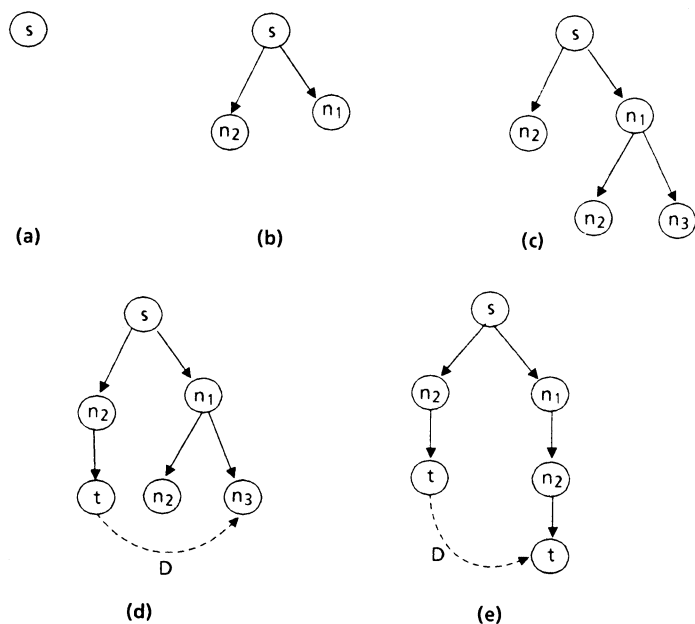


Figure 2. A directed graph  $G$ . Arc costs are given next to the arcs.

no larger cost than the extension of  $P''$ . Thus, the structure of the graph reveals dominance between two paths even when adequate bound information is not available. Note that the lower bound information may be used to conclude  $P'DP''$  (or  $P'DP'$ ) only if  $n$  is the terminal node. Figure 3 shows the operation of a best-first B&B using dominance for pruning on the graph of Figure 2.

The first two steps, as shown in Figures 3a, b, are identical to the ones shown in Figures 2a, b. Figure 3c shows  $A$  after splitting  $(s, n_1)$ . The path  $(s, n_2)$  dominates  $(s, n_1, n_2)$  because they both end at  $n_2$  and  $\text{cost}((s, n_2)) \leq \text{cost}((s, n_1, n_2))$ ; hence,  $(s, n_1, n_2)$  is pruned. In the next step  $(s, n_2)$  is chosen for splitting. As shown in Figure 3d, the resulting path  $(s, n_2, t)$  dominates  $(s, n_1, n_3)$ ; hence  $(s, n_1, n_3)$  is pruned. Now  $A$  contains only  $(s, n_2, t)$ , which is a singleton set. Therefore BB terminates with  $(s, n_2, t)$  as a shortest path between  $s$  and  $t$ .

**Complexity of B&B Search.** The complexity of a B&B search procedure depends on the problem being solved and the branching and pruning functions used. For NP-hard problems, the worst case complexity of any B&B procedure has to be exponential (assuming  $P \neq NP$ ), even though the B&B technique is usually much better than the exhaustive search. For example, for the  $n$ -city traveling-salesman problem, the worst case complexity of a B&B procedure given in Ref. 10 is  $O(2^n)$ , whereas the complexity of the exhaustive search procedure is  $O(n^n)$ . But even for NP-hard problems the average case complexity of a B&B procedure can be subexponential. Many researchers have tried to analyze the average complexity of the B&B procedures for specific abstract problem models. Dechter (13) developed a tree-based model of B&B and used it to investigate the relationships of the characteristics of the problem, the lower bound function, and the branching function to the average time complexity of the B&B procedure. Smith (14) analyzed the average time-and-space complexities for different search strategies used by special kinds of B&B procedures called relaxation-guided procedures. The complexity results of Huyn et al. (15) derived in the context of  $A^*$  (qv) are also applicable to B&B, as  $A^*$  can be viewed as a B&B procedure.

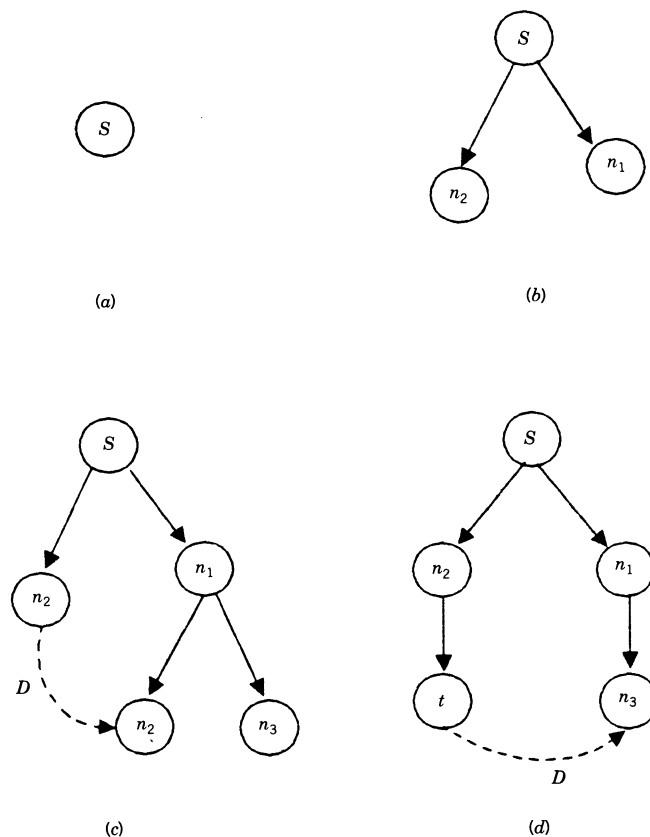


Figure 3. Steps of a best-first B&B search procedure using dominance for pruning.

### Relationship with AI Search Algorithms

The central idea of branching and pruning to discover an optimal element of a set is at the heart of many AI search algorithms. For example, the  $A^*$  algorithm (16) for state-space search can be viewed as a best-first B&B procedure (12).  $A^*$  differs only slightly from the best-first B&B procedure for finding a shortest path presented in this entry, as  $A^*$  uses a more informed lower bound. The nodes on the OPEN list in  $A^*$  represent the active set of paths on which branching and pruning is performed. The process of node selection and expansion corresponds to branching. Node elimination corresponds to pruning-dominated paths. Many other heuristic procedures for searching state-space graphs can be viewed as B&B procedures using different selection and pruning functions. The AND/OR graph search (qv) algorithm  $AO^*$  (16) and the game-tree (qv) search procedures alpha-beta (16), SSS\* (17), and  $B^*$  (18) can also be viewed as B&B procedures (7). In Ref. 19 it is shown that all these procedures can be viewed as instantiations of a general B&B procedure for searching AND/OR graphs (qv).

### Relationship with Dynamic Programming

B&B procedures are closely related to dynamic programming. Historically, the earlier dynamic programming procedures used "structural" dominance for pruning but did not use bounds. Selection was essentially breadth-first. Whereas, as noted before, the earlier formulations of B&B used bounds for selection and pruning but did not use dominance (in its general form) for pruning. In the recent formulations of dynamic programming (20,21) and B&B, both dominance and bounds

are used. Nevertheless, B&B and dynamic programming are different techniques for solving optimization problems. As discussed in Refs. 6 and 22, dynamic programming can be viewed as a bottom-up search, whereas B&B can be viewed as a top-down search (see Processing, bottom-up and top-down). In the context of state-space graphs, the difference between them vanishes, as for every state-space graph, it is possible to construct a dual state-space graph such that the top-down search of one is equivalent to the bottom-up search of the other (22). That is why Dijkstra's algorithm (23) for finding the shortest path in a graph could be classified both as B&B (24) and as dynamic programming (25).

## BIBLIOGRAPHY

1. E. Balas, "A note on the branch-and-bound principle," *Operat. Res.* **16**, 442-444, 886 (1968).
2. L. G. Mitten, "Branch and bound methods: General formulations and properties," *Operat. Res.* **18**, 23-34 (1970), Errata, *Operat. Res.* **19**, 550 (1971).
3. T. Ibaraki, "The power of dominance relations in branch and bound algorithms," *JACM* **24**, 264-279 (1977).
4. T. Ibaraki, "Branch-and-bound procedure and state-space representation of combinatorial optimization problems," *Inform. Ctrl.* **36**, 1-27 (1978).
5. W. H. Kohler and K. Steiglitz, "Characterization and theoretical comparison of branch and bound algorithms for permutation problems," *JACM* **21**, 140-156 (1974).
6. V. Kumar, A Unified Approach to Problem Solving Search Procedures, Ph.D. thesis, Department of Computer Science, University of Maryland, College Park, MD, December 1982.
7. V. Kumar and L. Kanal "A general branch and bound formulation for understanding and synthesizing and/or tree search procedures," *Artif. Intell.* **21**(1), 179-198 (1983).
8. E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Operat. Res.* **14**, 699-719 (1966).
9. T. Ibaraki, "Theoretical comparison of search strategies in branch and bound," *Inter. J. Comput. Inf. Sci.* **5**, 315-344 (1976).
10. Papadimitriou and Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
11. E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, Potomac, MD, 1978.
12. D. S. Nau, V. Kumar, and L. N. Kanal, "General branch-and-bound and its relation to A\* and AO\*," *Artif. Intell.* **23**, 29-58 (1984).
13. A. Dechter, A probabilistic Analysis of Branch-and-bound Search, Technical Report UCLA-ENG-CSL-8139, Cognitive Systems Laboratory, Computer Science Department, UCLA, Los Angeles, October 1981.
14. D. R. Smith, "Random trees and the analysis of branch and bound procedures," *JACM* **163**-188 (January 1984).
15. N. Huyn, R. Dechter, and J. Pearl, "Probabilistic analysis of the complexity of A\*," *Artif. Intell.* **15**, 241-254 (1980).
16. N. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, 1980.
17. G. C. Stockman, "A minimax algorithm better than alpha-beta?," *Artif. Intell.* **12**, 179-196 (1979).
18. H. Berliner, "The B\* tree search algorithm: A best-first proof procedure," *Artif. Intell.* **12**, 23-40 (1979).
19. V. Kumar, D. S. Nau, and L. N. Kanal, A General Paradigm for AND/OR Graph and Game Tree Search, AI TR 85-08, Artificial Intelligence Laboratory, University of Texas, Austin, TX, 1985.
20. T. L. Morin and R. D. Marsten, "Branch and bound strategies for dynamic programming," *Operat. Res.* **24**, 611-627 (1976).
21. V. Kumar, A General Bottom-up Procedure for Searching AND/OR Graphs, *Proceedings of the Fourth National Conference on Artificial Intelligence*, Austin, TX, 1984.
22. V. Kumar and L. N. Kanal, The Composite Decision Process: A Unifying Formulation for Heuristic Search, Dynamic Programming and Branch & Bound Procedures, *Proceedings of the Third National Conference on Artificial Intelligence*, Washington, DC, pp. 220-224, August 1983.
23. E. W. Dijkstra, "A note on two problems in connection with graphs," *Numer. Math.* **1**, 269-271 (1959).
24. P. A. V. Hall, Branch-and-Bound and Beyond, *Proceedings of the Second International Joint Conference on Artificial Intelligence*, London, pp. 641-658, 1971.
25. S. E. Dreyfus and A. M. Law, *The Art and Theory of Dynamic Programming*, Academic Press, New York, 1977.

V. KUMAR  
University of Texas

## SEARCH, DEPTH-FIRST

As stated in the entry on best-first search (qv), the task of solving a problem can often be formulated as the search for a path in a directed graph from an initial node to a goal node. The search begins by expanding the initial node, i.e., by generating its successors. At each next step one of the previously generated nodes is expanded until a goal node is found. There are many ways in which a generated node can be chosen for expansion, each having its own advantages and disadvantages. In depth-first search one of the most recently generated nodes is expanded first (1).

Figure 1 shows a tree-structured graph generated by a depth-first search procedure. Numbers on the left side of each node show the order in which the nodes are generated. Numbers on the right side of each node show the order in which the nodes are expanded. Since deeper nodes are expanded first, the search is called depth-first.

### Algorithm

The following is a depth-first search procedure under the simplifying assumption that the actual path from the initial node to the goal node is not needed.

1. Put the initial node on a list called OPEN.
2. If OPEN is empty, terminate with failure.
3. Remove the first node from OPEN. Call this node  $n$ .
4. If  $n$  is a goal node, terminate successfully.
5. If  $n$  has successors, generate all immediate successors of  $n$  and put them in an arbitrary order at the beginning of OPEN.
6. Go to step 2.

The procedure as presented above only finds a goal node. To find a path from the initial node to a goal node, step 5 of the procedure needs to be modified to establish a pointer from each generated node to  $n$ . The procedure also needs to keep those expanded nodes that are predecessors of the nodes on OPEN. These expanded nodes are kept on a list called CLOSED. So when a goal node is reached, the backward pointers can be traversed to find the path.



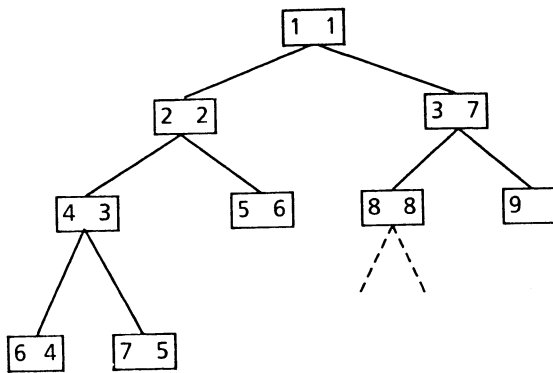


Figure 1. Search tree generated by a depth-first search procedure.

Depth-first search can be very effective if there is a goal node in the left part of the search tree. If the left part of the tree has infinite depth and has no goal node, the depth-first search would fail even if there is a goal node in the right side of the tree. To take care of such cases, the depth-first search procedure can be modified so that, in step 5, successors of  $n$  are generated only if the depth of  $n$  is no more than some constant  $L$ . This kind of search is called depth-bounded depth-first search. If there is no goal node at a depth  $L$  or earlier, the search would fail even if there is a goal node at a depth greater than  $L$ . In such cases the search will have to be restarted with a larger depth bound.

Usually, a depth-first-search procedure has lower storage requirement can be as much as  $O(k^n)$ . Depth-first search has  $k$  successors, the storage requirement of a depth-first procedure for searching to a depth of  $n$  is  $O(n^k)$ . In best-first search, if the heuristic evaluation function is bad, the storage requirement can be as much as  $O(k^n)$ . Depth-first search has very little overhead as compared to best-first search in which every generated node has to be evaluated and the OPEN list has to be rearranged after every node expansion. Hence, depth-first search is used quite frequently if good problem-specific evaluation functions are not available. Backtrack algorithms (2) (see Backtracking) are essentially depth-first search algorithms (3).

If the search space is a graph rather than a tree, a node may be generated more than once (4). The procedure as given would search the tree below a node each time it is generated. This duplication of work can be avoided if all the previously expanded nodes are kept on CLOSED. Now step 5 of the procedure can be modified such that only those immediate successors of  $n$  are put on OPEN that are not already in CLOSED or OPEN. But this modification increases the overhead and the storage requirement substantially.

For a more detailed discussion of depth-first search the reader is referred to Ref. 5.

## BIBLIOGRAPHY

1. A. Barr and E. A. Feigenbaum, *The Handbook of Artificial Intelligence*, Vol. 1, Kaufman, Los Altos, CA, 1981.
2. E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, Potomac, MD, 1978.
3. N. Nilsson, *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.
4. N. Nilsson, *Principles of Artificial Intelligence*, Tioga, Palo Alto, CA, 1980.

5. J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, Reading, MA, 1984.

V. KUMAR

University of Texas

SEGMENTATION. See Edge detection.

## SELF-REFERENCE

Although dreams of self-understanding systems have permeated AI since its earliest days, it has only been during the 1980s that self-reference has become a technical theme in AI programming languages, problem solvers, and reasoning systems. As the name suggests, self-reference involves two aspects. First, a self-referential system must have a prior ability to reason or refer. It must be about something; it must have a subject matter. Second, it must then be able to reason about or refer to, among other things, its self—either its self as a whole, or some aspect of itself, such as, e.g., its internal state, or location in the world. So a system that reasoned solely about medical diagnosis, or about blocks on a table, would not need self-reference. In contrast, one that deliberated not only about the placement of blocks but also about the completeness of its internal representations, would, in virtue of the latter ability, count as genuinely self-referential.

Self-reference has been thought powerful for a variety of reasons, partly as an economic way of giving systems a certain plasticity and increased functionality ("flexibility with elegance") and partly as a way of enabling them to cope with otherwise recalcitrant difficulties, particularly control problems arising from unconstrained search (qv) and inference (qv). For example, it is easy to imagine a system encountering some particular situation where the blind application of its normal procedures would lead it into serious combinatorial explosion (a system designed to cross-index all incoming facts, e.g., when presented with an online copy of this *Encyclopedia*). Such a system might be able to avoid these combinatorial problems if it were able, in advance, to assess the appropriateness of its default procedures to the circumstances at hand. Assessing the appropriateness of internal routines is a paradigmatically self-referential activity.

## History: Metalevel Reasoning and Control

In logic and philosophy of language, self-reference is a relatively narrow notion, typically used to describe a situation in which a single expression, sentence, or utterance refers to itself, such as *This very six-word noun phrase*, *This sentence is false*, or a person's utterance of *I'm lying*. The last two examples are famously paradoxical because they cannot consistently be said to be either true or false. Self-reference and related notions have provided important hygienic tests of foundational theories: They can reveal subtle inconsistencies and incoherences that challenge the integrity of formal accounts of logic, set theory, etc., on which semantics and mathematics depend. Three cases are of particular historical importance: Russell's (1) 1903 demonstration that Frege's proposal for a logical foundation for set theory was inconsistent, Gödel's (2) 1931 proof of the incompleteness of arithmetic (that there would always be true sentences of arithmetic that were not

provable, no matter what one's theory), and Turing's analysis of the halting problem (that there is no algorithmic way to decide, in general, whether a computer program will terminate). All three demonstrations involved examples with a self-referential twist, although even in these restricted and well-studied areas it is not precisely clear what self-reference is and how it relates to a variety of similar issues, such as diagonalization (Cantor's "trick"), self-applicability of a function [functions that are part of their own domain of applicability—i.e., licensing  $F(F)$ ], and self-exemplification of a property (properties, such as the property of being a property, that hold of themselves). A great deal of work in mathematics and logic in this century has focused on developing systems that are flexible and powerful, and even permit various sorts of limited self-reference (in this narrow sense), without foundering on the puzzles and paradoxes of which this narrow sort of self-reference is a reliable source.

The important notions of self-reference in AI, however, are of a much broader sort than illustrated in the above examples: they have to do with whole systems, not individual expressions or symbols, and with agency and process, not just static or abstract structure. They have to do, that is, with the lay, intuitive notion of "self" in the sense that one might say of someone that he/she does not have much self-knowledge. Results so far have been modest; AI has certainly not yet developed anything like a theory of the notion of "self" that is claimed to have emerged during the last several hundred years in the West. But it is toward that substantial notion of self that AI research is implicitly directed.

AI's interest in the self emerged in part as a response to complexity, particularly the control complexity that arises from the undirected use of declarative representations of general facts. In the relatively "early" days of AI (primarily the late 1960s and early 1970s) a "procedural" orientation toward knowledge representation (qv) was widely espoused, especially by adherents to the MIT school. This procedural orientation had arisen in reaction to the generally recognized failure of totally domain-independent general inference systems (such as A. Newell and H. Simon's GPS (3)). The proceduralists argued that in place of general undirected theorem proving (qv) or goal satisfaction the way to make a system behave intelligently was to encode, in situation-specific procedures, specific recipes or algorithms for behavior. So, rather than telling such a system general facts about, say, the world of car repair, the idea was to discern in advance what actions such a system would need to take and to program them up in advance (say, by coding in algorithms for muffler repair or for adjusting the timing chain). Winograd's SHRDLU (qv) system (4) for handling natural-language discourse about the blocks world was perhaps the most famous system in the proceduralist style. But this paradigm, too, eventually proved weak, primarily because the resulting systems were untenably brittle and non-modular: They could not cope with novel situations, were hard to maintain or modify, and often required complete reengineering when the repertoire of desired responses was even minimally extended.

So the proceduralist approach in turn generated its own reaction. The response involved a shift to a so-called knowledge-intensive approach, in which designers gave systems the requisite specificity by encoding domain-specific knowledge in the form of declarative "rules," which represented the relevant facts about the given domain but were not structured in ad-

vance to suit any particular line of activity or reasoning (see Rule-based systems). These databases—or "knowledge bases," as they are now called in the world of expert systems (qv)—were coupled with a general inference (qv) or deduction engine, whose goal it was to examine the representations of the facts germane to a particular problem and to deduce or infer from them an acceptable solution or plan of activity. The approach puts great demands on the formal scheme chosen for representing all this domain-specific knowledge. Computational encodings of first-order logic (qv) became the most popular choice of representational medium, leading naturally to the use of inference systems effecting some variant of logical deduction.

But there was (and is) a problem with such systems: The deductive relation—what can be concluded from a premise  $P$ —is wildly open-ended, leading to serious problems of combinatorial explosion in the control structure. Without some further narrowing or specification of what was to be concluded, such systems tend to wander forever in unconstrained search. The problem is that one cannot construct effective systems merely by representing what such systems should know, paying no attention to how they should use that knowledge. What was needed was some way to control the deductive or inference procedure explicitly, at the same time retaining the apparent modularity and flexibility of the declarative representational scheme.

It does not take a very complex example to see how such troubles arise. Suppose one wants to know whether Einstein's parents were American. One possibility would be to enumerate all Americans and see for each whether he or she was a parent of the famous physicist. But it would be much more efficient to enumerate Einstein's parents first and then to see whether each of them was American. On the other hand, if one wanted to know whether he/she has ever met a pope, the better choice would be to enumerate the popes, and ask oneself, for each one, whether one has ever met him, rather than attempting to enumerate everyone one has ever met, and asking whether each was a pope. The point is that although the logical structure of the two questions is the same (i.e., for some known  $X$ , does any  $Y$  related to  $X$  by relation  $R$  possess property  $P$ ?), facts about the size of the sets involved impinge on the tactics of how to look for an answer. If these were only questions to be asked, one might encode the strategies right along with the basic factual representation (this was the proceduralists' strategy), but then one loses all modularity and all ability to use the same facts to answer different questions.

So a method was needed that would enable strategies to be decided intelligently, based on a variety of factors (including the structure and content of both facts and questions), without sacrificing the underlying declarative representational stance. The first move was to make the control information explicit, as discussed and advocated, e.g., in Refs. 5–7. This requires two things: a language in which to represent the control information and a theory of control in terms of which to formulate those representations. Both issues are familiar, for they are exactly the same as the issues involved in representing the main body of declarative facts about the primary subject matter. This leads naturally to the second move—the seminal idea behind all extant self-referential systems in AI. The idea was to treat the problem of controlling the inference procedure as strictly analogous to any other problem to be solved and therefore (this does not quite follow directly, but it is the natural

idea) to represent the relevant facts about deduction and inference in the same language as the facts about the original problem domain—medicine, investment, blocks, whatever.

The idea, in other words, was to give the system so-called metalevel representations—“meta” because they were about the basic (object-level) representations (see Metaknowledge, metarules, and metareasoning). Of course, the deductive problem recurses; it remains unclear how to constrain the search space for the inference procedures that are intended to use these metalevel representations to reason about control. But the basic intuition, seemingly at least partially borne out in practice, was that this iterated problem would be simpler than the original problem in some way that would allow it to be treated tractably. There are two reasons for this optimism. First, it was hoped that the problem of controlling inference would be smaller than more general real-world problems (fewer principles to apply; fewer data to examine). Second, and more important, it was hoped that the control problem would remain essentially the same across different object-level domains—i.e., that the same metalevel representations would be applicable even while the object-level representations changed in content. Whereas the object-level representations would vary as widely as the domains for which the overall systems were designed, the metalevel representations would always be about rationally controlled inference, which might require special-purpose control constructs, but they would always be the same. For example, one might expect metalevel rules of the following generic sort: use information from the smallest enclosing statistical sample, reason forward from specific rather than general facts, employ depth-first search only in nonrecursive domains, etc.

This idea of metalevel control has been advertised since the late 1970s, as is evident from such papers Refs. 8–10. The increasing role of logic as a representational and inferential medium, mentioned above, led in turn to the adoption of metalevel control in a variety of logic-based deduction systems, of which Weyhrauch's FOL (9), a full problem solver, and Bowen and Kowalski's metalevel-controlled logic-programming language (11) are the best known. Davis (12) provides a more recent analysis of the role of metalevel rules for controlling AI systems; Genesereth et al.'s MRS (13) is a recent system based very explicitly on the same idea.

### Varieties of Self-Reference

Although the use of metalevel rules to control inference is historically important to AI's concept of self-reference, it is not the concept itself. In order to give some coherence to the overall subject of self-reference, Smith (14) distinguishes four ways in which a (computational) agent may be relevant to its conception of some subject matter.

1. *Indexical*: An agent's representations or beliefs are called *indexical* or *self-relative* just in case the way in which the agent conceives of its subject matter depends on some aspect of the agent, although not (necessarily) in a way that the agent makes explicit to itself. For example, in a simple case of saying that a man is to the right of a tree, an agent is liable to employ an indexical two-place relation “to-the-right-of,” leaving the requisite third component of the full relation implicit, presumably relative to its own position and orientation. (People are so used to this kind of indexi-

cality that they often forget that “to-the-right-of” is in fact at least a three-place relation.)

2. *Autonymic*: Any system that “knows” its own name, in some appropriate sense of “know,” is labeled *autonymic*. The name need not be proper or unique; rather, what is required is that the system respond differentially, in some nontrivial way, to some term that refers to it. For example, systems that process electronic mail are typically able to recognize (and deal specially with) messages addressed to their own users, forwarding other messages to neighboring machines. Though modest, such a capability will be counted as autonymic.
3. *Introspective*: An agent's *introspective* abilities are those that enable it to obtain (typically privileged) access to its own internal structures, operations, and behavioral potential in such a way as to matter to its present and future actions. Thus, e.g., a system that, in the midst of a computation, was able to report on what it is doing, receives guidance about how to change or affect its strategies, and then continues the computation in line with the new advice must have a nontrivial introspective capability.
4. *Reflective*: To be *reflective*, a system must be able to reason or deliberate not (only) about its internal structure or state but also about its situation and embedding context. Thus, whereas knowing that one was repeating oneself would require only introspection, it would require reflection to realize that by doing so one was acting inappropriately.

The incorporation into a computational system of capabilities of each of these four types requires rather different sorts of mechanisms (see below).

**Indexicality.** To develop a sense for indexicality, it is instructive to look at English (or any natural language) because it is so extraordinarily indexical. Indexical interpretations are obviously required not only for such terms as the pronoun “I,” the demonstrative “that,” etc., but also for tensed verbs (which is to say, for essentially all verbs—past tense, e.g., is used to describe an event that happened before the time of utterance, which is a fact about the speaker) and other grammatical forms and for a variety of such apparently unambiguous nouns as some proper names and times (e.g., on the West Coast, one typically says that it is 9:00, not that it is 9:00 PDT (Pacific daylight time), and assumes that the interpretation of the shorter phrase will be indexed relative to the circumstances—specifically, in this case, to the location). In fact, it has been suggested that all natural language is indexical: even the phrase “2 plus 2 equals 4,” about as noncontextual a thing as it is possible to say, is stated in the present tense; it requires inference to recognize that the proposition being thereby expressed is a contextually independent eternal verity. Similarly, even when pseudological representations are used internally in AI systems, indexical interpretations are typically employed: the interpretation of an internal representation  $\text{RIGHT}(\text{MAN}_5, \text{TREE}_{14})$ , e.g., to use the example mentioned above, is not only contextually relative but also contextually relative in the particular way demanded by indexicality: The implicit but requisite third axis is defined by the location of the agent bearing the representation. Logic per se does not deal with indexicality, but, as this example shows, computational representations based on logical language can nonetheless be used in indexical ways.

Two things about indexicality are important for AI. First, almost all AI systems are already indexical to a great extent, as the previous examples should suggest. Second, indexicality is largely a semantic fact, not an architectural fact; one should not expect to build or find mechanisms of indexicality (mechanisms that add indexicality to an otherwise nonindexical system as if it were an add-on property)—in fact, it is not clear that there could be any such thing. Rather, to admit that a system is indexical is to note how contextually relative reference typically is—relative, among other things, to facts of the system's overall circumstance. Just as it would be a category error to assume that a theory of reference must be represented within a system in order to provide that system with referential capabilities, it would similarly be a category error to assume that either mechanisms of indexicality, or representations of a theory of indexicality, would be required in order to render a system indexical. Indexicality just is; it is for theorists to notice, not for designers to implement.

**Autonymy.** About autonymic systems not much need be said, except that from such an ability there is no reason to suppose that the agent has anything substantial to say about itself—i.e., this is not in and of itself a theory-relative capacity. All that is required is that the name play a distinguished role, implying that the agent's behavior depend on the fact that its own name is its own name. It is easy enough to imagine, in contrast, an expert system designed to diagnose possible hardware faults based on statistics of recoverable errors. Such a system might be given data on its own recoverable errors, filed under a name known to its users to refer to it. But the system's behavior would not be any different in this case; it would yield its conclusions entirely unaffected by the self-referential character of this externally attributed reference. When a system or agent responds differentially to its own name, on the other hand—as, e.g., when a mail system recognizes its own users' mail—it is called autonymic. As with indexicality, the autonymic character of a system may be more or less central to its structure or behavior.

**Introspection.** It is on introspective abilities that most recent research on self-reference has focused, so this aspect of self-reference is surveyed in the greatest depth. The goal here is to produce systems with the ability to examine, manipulate, and draw conclusions about their internal structure and state as well as about their primary (and presumably external) domain. The use of metalevel rules to reason about control, discussed earlier, falls squarely into this area, as do most of the other systems already cited: FOL (qv), MRS, etc. It is striking, however, that actually defining computational introspection rigorously enough to demonstrate that these systems (and not all AI systems) are introspective is beyond the current state of the theoretic art. The problem is that all computer systems—indeed, all state machines—behave in ways that depend in various ways on their internal structure and state. Rather, what is distinctive about introspective systems is that they reason not just with their internal structures but about their internal structures or state. And aboutness—the “reference” part of the two ingredients in self-reference—is not yet a theoretically reconstructed notion in AI (in spite of its being, at least on this author's view, the defining aspect of computation).

One way in which introspective capabilities arise in representation-based systems, again as suggested by the example of metarules for control, is in representational structures manifestly about other (object-level) representational structures. But, as seen even in the case of controlling inference about parents and popes, the subject matter of introspective structures and deliberations can include not only the representations that are internal to the system but also other less “objectlike” aspects of the system, including the use of the representations, and other activities in which the system engages. Furthermore, all these aspects become the subject matter of introspective deliberation by being represented in a further set of internal structures. And—a crucial point—this requires that a theory of internal operations be implicitly adopted in order to formulate this explicit account of what might otherwise have remained implicit. Furthermore, these theory-relative metalevel representations must play a causal role in the behavior of the system in which they occur if the introspective abilities are to come to anything substantial. Three characteristics, in sum, are required of an introspective agent: referential abilities, a theory-relative representation of self, and appropriate causal connection.

Smith's work on what he called “reflection” in LISP (15) (but according to the present framework it was misnamed—it was, rather, an account of introspection in programming languages) attempted to deal explicitly with all three of these aspects. Specifically, he defined a notion of reference by introducing a two-factor semantical account, distinguishing the “procedural” semantics (what the system did) from the “referential” semantics (what the system was about); embedded a theory of the system within the system in terms of which the metalevel structures were formulated; and proposed a particular mechanism whereby the requisite causal connection could be established between the implicit operations of the agent on the one hand and explicit introspective representations of those operations on the other. Without such causal connection, a system could not count as introspective but would merely be able to model itself in as completely disconnected a way as the fault-diagnosing example system mentioned above was able to reason about itself. It is a universal assumption of those aiming to produce introspective systems that the introspective abilities must matter to the system—enabling it to shift back and forth, in a flexible and consequential way, between “thinking” about its primary domain and “thinking” about itself.

Different models of reference, different theories of self, and different mechanisms for causal connection have been adopted in various other systems cited. Weyhrauch, in his work on FOL (9), e.g., employs two languages (one based on logic, one based on LISP (qv)), defines both reference and causal connection as relationships between these two languages, and uses reflection principles from logic to support the requisite causal connection. Systems also differ on which introspective abilities are self-applicable (i.e., whether the system can introspect about its introspective abilities). MRS, e.g., is based on a single-level introspective ability; Smith's 3-LISP and Weyhrauch's FOL are designed to introspect to an arbitrary degree. And, finally, the various systems that have been proposed differ considerably on the theory that must be (explicitly or implicitly) adopted, in terms of which the internal operations and structures are characterized. Introspection becomes

complex, and correspondingly powerful, when there is substantial implicit state to the inferential procedures that must be represented explicitly in introspective representations.

The move from state to representation of that state—required for appropriate causal connection—is a theme that permeates all models of introspection and underlies much of their power. The point is simple in principle, although its proper implementation can involve extraordinary intricacy. What is typically required are facilities to render explicit what was otherwise implicit. For example, whether a particular rule or program is used more than once in a system would normally be an implicit, or relational, fact: one that is true in virtue of various facts and properties of objects scattered throughout the system but not explicitly represented—as it would be, e.g., if somewhere in the system one found the representation Times-Used(Program<sub>17</sub>,3), providing that had the appropriate meaning. If, on the other hand, the introspective theory (namely, that theory with respect to which the introspective abilities are defined) included “number of times used” as one of its theoretic relations, then an act of introspection about Program<sub>17</sub> would have to create that explicit sentence.

Constructing explicit representations of implicit facts is important because, in computer systems, what is explicit is local and what is local can be reacted to directly. In general, one of the most important aspects of introspection is that it provides a means for making local what is otherwise implicit and/or global, thereby setting it up for explicit treatment. However, and this is where causal connection gets its bite, someone must guarantee that the (implicit) genuine facts of the matter (how many times Program<sub>17</sub> is, in fact, used) and the explicit representation of that fact (as recorded in Times-Used(Program<sub>17</sub>,3)) stay synchronized. In particular, it is important that the introspectively accessible explicit representation be true. But that fact does not decide a further question: Which one is intended to follow the other? Since truth is an asymmetric relation, one might expect that the implicit fact (the number of times the program is actually called) would be the dominant situation, to which the explicit representation should be subservient. And indeed this is often the case. But it is also common, in designing introspective systems, to want the causal connection to “go the other way around.” For example, suppose some system contains an introspectively accessible representation saying that the system will pursue a particular goal for 10 minutes before stopping to assess the situation. And suppose further that, while introspecting, the system decides to change that representation to say that it will pursue a goal for only 5 minutes between pauses. What one probably wants is for the underlying behavior of the system to change in virtue of that change in description. That is, in this case it is the description that is assumed to be dominant; the described reality, to be subservient. In other words, as well as guaranteeing the integrity of the reality-to-description link between implicit fact and explicit representation of that fact, one also wants the system to provide a description-to-reality link, so that introspective deliberations about how one would like the system to be can take effect. Both directions are crucial to the “mattering” claimed as necessary in order to say of a system that it was genuinely introspective.

As even these simple examples suggest, providing causal connection is one of the most subtle issues faced by the designer of an introspective system. On the other hand, rigorous

theoretical vocabulary in which to understand the salient issues has not yet been proposed. And of course none of the terms used here—“implicit,” “explicit,” “local,” “global”—are any more theoretically solid than “introspection” or “self-reference.” What these considerations indicate are the sorts of issues with which a genuine theory of introspection would have to grapple. Batali (16) begins a comparative analysis, but much more work is yet to be done.

**Reflection.** At the time of writing (1985), no one has yet presented a system that counts as genuinely reflective under the characterization presented above. To the extent that substantial self-reference has been achieved, it remains of an introspective nature—an agent’s representing itself to itself, without the perspective or detachment that would come from its considering not only its internal state but its entire situation, including its embedding context. Of the various reasons for this failure, the following is probably the most substantial: the semantic relationship between agents and their embedding situation, as noted above with respect to reference in general, has yet to be theoretically reconstructed and used in the characterizations with which one designs and analyzes the systems constructed. The current theoretic techniques of AI are at present largely “internalist,” focusing solely on the structure, interrelationship, and behavioral consequences of internal representations. Recently, however, a “neo-realist” methodological stance is beginning to affect semantical accounts in a variety of fields, including AI. This realist stance is particularly represented in research stemming from Palo Alto and the Center for the Study of Language and Information, for which the term “situated” has become a common identifying epithet. See, e.g., Barwise and Perry’s situation semantics (17), Rosenschein and Pereira’s situated automata (18), and Suchman’s accounts of situated action (19). These new accounts offer the promise of more adequate theoretical foundations in terms of which to define genuinely reflective systems, or at least to tackle the problems involved, of which the ability to represent implicit indexical beliefs in explicit nonindexical ways is a salient example.

### Self-Reference and Consciousness

Inevitably, any informal discussion of self-reference in AI seems to end with a discussion of consciousness—that enduring, if ill-understood, aspect of human phenomenal experience. Considerations of consciousness have not, however, arisen in the technical discussions of self-reference surveyed in this brief sketch. From that fact alone one cannot conclude that the technical work does not bear on the larger subject, partly because AI systems themselves are often more interesting and subtle than can be discerned from the way they are theoretically described. Buried in the architectural details and technical solutions of some of the systems cited in this entry are more distinctions, choices, and insights than can be adequately described here, not just for reasons of space but because of fundamental deficiencies in the present state of theory. Reconstructing those architectural aspects and developing increasingly sophisticated introspective and reflective capabilities will occupy AI researchers for the indefinite future. Whether they point in a direction that will eventually lead to an understanding of human consciousness is hard to say, although there is

some reason to think they have more to do with self-consciousness than with the simpler, but more important, notion. The best conclusion to be drawn at the moment is merely this: Self-reference is not a technique or a simple property that can be added to systems. It is a large and complex issue—one that will not be fully understood for a very long while.

## BIBLIOGRAPHY

1. B. Russell, letter to Frege of June 16, 1902, reprinted in J. van Heijenoort, ed., *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*, 124–125, Harvard University Press, Cambridge, 1967. For a fuller treatment see B. Russell, *The Principles of Mathematics*, 1, Cambridge University Press, Cambridge, UK, 1903; 2nd ed., Norton, New York, 1964.
2. K. Gödel, "Über formal unentscheidbare Sätze der Principia mathematica und verwandter Systeme I," *Monatshefte für Mathematik und Physik* 38, 173–198; English translation: K. Gödel, "On formally undecidable propositions of *Principia mathematica* and related systems I," in J. van Heijenoort, ed., *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*, 596–616, Harvard University Press, Cambridge, MA, 1967.
3. G. W. Ernst and A. Newell, *GPS: A Case Study in Generality and Problem Solving*, Academic Press, New York, 1969.
4. T. Winograd, *Understanding Natural Language*, Academic Press, New York, 1972.
5. J. de Kleer, J. Doyle, G. L. Steele, and G. J. Sussman, "Explicit Control of Reasoning," in *Artificial Intelligence: An MIT Perspective*, ed. P. H. Winston and R. H. Brown, MIT Press, Cambridge, MA, 1, 93–118, 1979. Also in: *Proceedings of the Symposium on Artificial Intelligence and Programming Languages*, 1977, and *Readings in Knowledge Representation*, R. J. Brachman and H. J. Levesque, eds., Morgan Kaufman, Los Altos, CA, 345–355, 1985.
6. R. Kowalski, "Algorithm = Logic + Control," *CACM* 22, 424–436 (July 1979).
7. J. McCarthy and P. J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence," in *Machine Intelligence 4*, Bernard Melzer and Donald Michie, eds., Edinburgh University Press, Edinburgh, UK, 463–502, 1969.
8. R. Davis, "Applications of meta level knowledge to the construction, maintenance, and use of large knowledge bases," Stanford AI Memo 283 (July 1976), reprinted in R. Davis and D. B. Lenat (eds.) *Knowledge-Based Systems in Artificial Intelligence*, McGraw-Hill, New York, 229–490, 1982.
9. R. Weyrauch, "Prolegomena to a Theory of Formal Reasoning," *Artificial Intelligence*, 13(1,2), 133–170, 1980.
10. J. Doyle, "A Model for Deliberation, Action, and Introspection," Massachusetts Institute of Technology, Cambridge, MA, Artificial Intelligence Laboratory Memo AIM-TR-581, 1980.
11. K. Bowen and R. Kowalski, "Amalgamating Language and Metalanguage in Logic Programming," in *Logic Programming*, ed. K. L. Clark and S.-A. Tarlund, Academic Press, New York, 153–172, 1982.
12. R. Davis, "Meta-Rules: Reasoning About Control," *Artificial Intelligence* 15(3), 179–222 (December 1980).
13. M. R. Genesereth, R. Greiner, and D. E. Smith, "MRS—A Meta-level Representation System," HPP-83-27, Stanford University Heuristic Programming Project, 1983.
14. B. C. Smith, "Varieties of Self-Reference," in *Theoretical Aspects of Reasoning About Knowledge: Proceedings of the 1986 Conference*, Los Altos, CA. Morgan Kaufmann, 19–43, 1986.
15. B. C. Smith, "Reflection and Semantics in LISP," *Proceedings of the ACM Principles of Programming Languages Conference*, Salt Lake City, UT, 23–25, 1984.
16. J. Batali, "Computational Introspection," MIT AI Laboratory Memo, AIM-701, Cambridge, MA, 1983.
17. J. Barwise and J. Perry, *Situations and Attitudes*, Bradford Books, Cambridge, UK, 1983.
18. S. J. Rosenchein, "Formal Theories of Knowledge in AI and Robotics," *New Generation Computing*, 3(4), Ohmsha, Ltd., Tokyo, Japan.
19. L. Suchman, "Problems in Human-Machine Interaction," Cambridge University Press, Cambridge, UK, 1987.
20. B. C. Smith, "The Correspondence Continuum," *Proc. of the Sixth Canadian Conference on Artificial Intelligence*, Montreal, Canada, 1986.

B. C. SMITH

Xerox PARC and Stanford University

## SELF-REPLICATION

In machine self-reproduction an instruction-obeying device (such as a general-purpose computer) is augmented with physical manipulation capability (as in an industrial robot), supplied with raw materials, and programmed to produce a duplicate of itself. A theoretical model of this process was proposed by von Neumann (1) in which the initial machine resides in an environment of spare parts (switching, sensing, cutting, fusing elements, etc.). The parent machine plucks parts at random from its surroundings, identifies them, and following stored instructions, assembles the parts into a duplicate of itself.

This informally described kinematic model was superseded by von Neumann's cell-space model (2–4). [Conway's "Game of Life" is an example of an extremely simple cell-space system (5).] In von Neumann's cell-space model machine reproduction takes place in an indefinitely extended, two-dimensional rectangular array, each square of which contains an identical automaton in direct communication with its four cardinal-direction neighbors. Each cell automaton is capable of being in any one of 29 different states. These states determine the way in which a cell automaton interacts with its neighbors. Depending on its state and the state of its neighbors, a cell automaton can transmit, switch, or store information or can undergo a change of state. Configurations of cell automata can be designed to form higher order information-processing devices, such as pulsers (units which when stimulated emit a stream of pulses) and decoders (units activated only upon receipt of particular patterns of pulses). These and other higher order units can be combined to form a self-reproducing machine consisting of a general-purpose computer with an indefinitely expandable memory unit and a constructor (a device containing banks of pulsers that can emit signals that cause a cell automaton to assume any one of the 29 states).

The self-reproducing process proceeds as follows: The parent machine, reading instructions from memory, first directs the constructor to produce trains of pulses that transform cell automata at the periphery of the original machine, so that a constructing-arm pathway of newly activated cells is created and extended out into an undifferentiated region of the cell space. Then the parent machine, making use of a stored description of itself, directs the arm to move and to emit pulses so as to produce a configuration of cells that is identical to that of the original machine (though as yet lacking the memory contents of the original). The parent machine then reads its memory a second time and loads a copy of the contents into the



memory of the offspring machine, turns on the new machine, and withdraws the constructing arm. This completes the self-reproduction.

This process of self-reproduction thus has two principal phases: first, the memory unit contents are read and interpreted as instructions for construction; next, the memory is read a second time in order to load a copy into the new memory. This action parallels the biological process of reading nucleic acids twice, once to carry out protein synthesis and again to replicate the genetic message.

Theoretical research since von Neumann has taken several directions. Alternative (usually simpler) cell spaces have been shown capable of supporting the reproductive process (6,7). Hybrid cellular-kinematic systems have been devised that make machine movement a more direct process (8) (in the original von Neumann cell space, machine movement is implemented by erasing a configuration of cells in one location and recreating it in another).

Other hybrid systems emphasize machine capacity for identifying system componentry (9). In such systems a machine may initially possess less than complete knowledge of itself but may still be able to reproduce itself since the deficiency can be made up by self-inspection (10). This also means that a machine can undertake partial self-repair: The machine compares its present configuration (obtained through self-inspection) with what its configuration should be (as contained in a stored description of itself) and uses its constructor to reduce the discrepancy. This strategy can be generalized to enable robotic machines to exhibit intentional goal seeking and evolution (11).

In a machine evolutionary process successive offspring machines cannot be mere exact duplicates of parents but must in some respect come to be both different and superior. One approach to machine evolution is to mimic the natural evolutionary processes of random variation of type and subsequent selection of better adapted types, but Myhill has shown that an indefinitely continued sequence of reproducing machines, each offspring superior to its parent, can be produced in an entirely deterministic fashion (12).

Machines that accept inputs and produce outputs can be viewed as implementing mathematical functions. In self-reproduction machines read or otherwise refer to or act upon themselves to produce their outputs. Recursive-function theory is the abstract and general study of such self-reference computations and is thus an important tool for the precise investigation of self-reproducing systems. For example, the results and techniques of recursive-function theory were employed in the Myhill result cited above and also in establishing the conditions under which a reproducing machine system will eventually have a sterile descendant, will continue to produce descendants indefinitely in a periodic manner, or will produce descendants indefinitely but aperiodically (13).

The processes by which artificial machines can exhibit various forms of reproduction, self-inspection, repair, and evolution can serve as explanatory models of similar processes in natural biological systems as well as contributing to the development of a broad theoretical biology of the possible organisms of possible universes.

Though complete physical artificial-machine self-reproduction has not yet been achieved, automation in which computer-controlled machines carry out the manufacture of other machines (including computing machines) has been moving steadily in that direction. Full exploitation of the concept will

take advantage of the exponential nature of the reproductive process. Environmental concerns and the cost of energy and raw materials severely constrain Earth-based manufacturing of such an explosive nature. The lunar surface has been proposed as a suitable environment for the first economically practicable general-purpose self-reproducing factory (14).

## BIBLIOGRAPHY

1. J. von Neumann, The General and Logical Theory of Automata, in L. A. Jeffress (ed.), *Cerebral Mechanisms in Behavior*, Wiley, New York, pp. 1-31, 1951.
2. J. von Neumann, *Theory of Self-Reproducing Automata*, edited and completed by A. W. Burks, University of Illinois Press, Urbana, IL, 1966.
3. A. W. Burks, Von Neumann's Self-Reproducing Automata, in A. W. Burks (ed.), *Essays on Cellular Automata*, University of Illinois Press, Urbana, IL, pp. 3-64, 1970.
4. J. Thatcher, Universality in the von Neumann Cellular Model, in A. W. Burks (ed.), *Essays on Cellular Automata*, University of Illinois Press, Urbana, IL, pp. 132-186, 1970.
5. M. Gardner, "Mathematical games," *Sci. Am.* **224**, 112-117 (1971).
6. E. Codd, *Cellular Automata*, Academic Press, New York, 1968.
7. E. R. Banks, Universality in Cellular Automata, *Proceedings of the Eleventh Switching and Automata Theory Conference*, pp. 194-215, 1970.
8. M. Arbib, "Simple self-reproducing universal automata," *Inf. Ctrl.* **9**, 177-189 (1966).
9. R. Laing, "Some alternative reproductive strategies in artificial molecular machines," *J. Theoret. Biol.* **54**, 63-84 (1975).
10. R. Laing, "Automaton models of reproduction by self-inspection," *J. Theoret. Biol.* **66**, 437-456 (1977).
11. A. W. Burks, Computers, Control, and Intentionality, in D. Kerr et al. (eds.), *Science, Computers, and the Information Onslaught*, Academic Press, New York, pp. 29-55, 1984.
12. J. Myhill, The Abstract Theory of Self-Reproduction, in A. W. Burks (ed.), *Essays in Cellular Automata*, University of Illinois Press, Urbana, IL, pp. 206-218, 1974.
13. J. Case, "Periodicity in generations of automata," *Math. Syst. Theor.* **8**, 15-32 (1974).
14. R. Cliff, R. Freitas, R. Laing, and G. von Tiesenhausen, Replicating Systems Concepts: Self-Replicating Lunar Factory and Demonstration, in R. Freitas and W. P. Gilbreath (eds.), *Advanced Automation for Space Missions*, NASA/ASEE Conference, Santa Clara, CA, Publication 2255, pp. 189-335, 1980.

R. LAING  
Consultant

## SEMANTIC NETWORKS

A semantic network or net is a structure for representing knowledge as a pattern of interconnected nodes and arcs. The earliest semantic networks were designed as intermediate languages for machine translation (qv), and most versions are still strongly oriented toward the features of natural languages. But the more recent versions have grown in power and flexibility to compete with frame systems (see Frame theory) and logic programming (qv) systems as general knowledge representation languages.

Since the late 1950s dozens of different versions of semantic networks have been proposed and implemented. Because of the diversity, the terminology and notations vary widely. Certain themes, however, are common to most versions such as:

nodes in the net represent concepts of entities, attributes, events, and states;

different nodes of the same concept type refer to different individuals of that type, unless they are marked with a name, identifier, or coreference link to indicate the same individual;

arcs in the net, called conceptual relations, represent relationships that hold between the concept nodes (labels on the arcs specify the relation types);

some conceptual relations represent linguistic cases, such as agent, object, recipient, and instrument (others represent spatial, temporal, logical, and intersentential connectives);

concept types are organized in a hierarchy according to levels of generality, such as ENTITY, LIVING-THING, ANIMAL, CARNIVORE, FELINE, CAT; and

relationships that hold for all concepts of a given type are inherited through the hierarchy by all subtypes.

Besides these commonalities, the various networks diverge on a number of issues: philosophical questions of meaning; methods for representing all the quantifiers and operators of symbolic logic (qv); techniques for manipulating the networks and drawing inferences (qv); and notations and terminology that differ from one author to another. Despite the differences, all the versions are based on some common assumptions: network notations are easy for people to read, efficient for computers to process, and powerful enough to represent the semantics of natural languages.

### Historical Survey

The first complete system of first-order logic used a graph notation: Frege's *Begriffsschrift* (1) represented formulas as tree structures, but they did not resemble modern semantic nets. Another pioneer in logic, Charles Sanders Peirce, was impressed with graph notations in organic chemistry. In 1897 he abandoned linear notations for logic in favor of his new existential graphs. Peirce defined elegant rules of inference for the graphs and extended them beyond first-order logic to modal and higher order logic. He even stated all the axioms for existential graphs in existential graphs themselves. Unlike Frege's *Begriffsschrift*, Peirce's existential graphs have important implications for semantic nets: They are similar to Hendrix's partitioned nets (2,3), and Sowa (4) adopted them as the logical basis for his conceptual graphs. See Roberts (5) for a systematic presentation of Peirce's graphs.

In psychology, Selz (6,7) used graphs to represent patterns of concepts and the inheritance of properties in a concept hierarchy. His theory of schematic anticipation had strong similarities to AI theories of frames and pattern-directed invocation of procedures. Selz's theories had a strong influence on de Groot's studies of chess playing (8) (see also Computer chess methods). De Groot, in turn, influenced Newell and Simon's work on problem solving (qv). Their student Quillian developed a network system for semantic memory (qv) and cited

Selz in his dissertation (9). Frijda and de Groot (10) present a summary of Selz's work. See Norman and Rumelhart (11) and Anderson and Bower (12) for later applications of semantic networks in psychology.

In linguistics Tesnière was developing graph notations in the 1930s for his system of dependency grammar. In the United States Hays (13,14) and Klein and Simmons (15) adopted dependency theory for computational linguistics (qv). They influenced Schank, who shifted the emphasis from syntactic dependencies to conceptual dependencies (qv) (16,17). Tesnière has had a major influence on European linguistics. Valency theory (18), which is used in some machine translation systems (19), is also derived from his work. Tesnière's posthumously published book (20) is still worth reading for its linguistic insights and analyses.

The first implementations of semantic networks were developed for machine translation systems in the late 1950s and early 1960s. In fact, the first system to be called a semantic network was Masterman's version at Cambridge University (21). She had developed a set of 100 primitive concept types, such as FOLK, STUFF, THING, DO, and BE. In terms of these primitives her group defined a conceptual dictionary of 15,000 entries. She organized the concept types into a lattice and had a mechanism for inheriting properties from supertypes to subtypes. Wilks, who is best known for his theory of preference semantics (22,23), continued to use Masterman's primitives as a basis for his work. Another system for machine translation was based on Ceccato's correlational nets (24,25). Ceccato defined a list of 56 different relations, which included case relations, subtype, member, part-whole, and miscellaneous relations such as kinship. He used the nets of concepts and relations as patterns for guiding a parser and resolving syntactic ambiguities. In 1961 an important conference on machine translation was held at the National Physical Laboratory in England: Ceccato, Hays, and Masterman presented papers on their networks; and other early researchers, such as Quillian, were on the attendance list.

In other AI systems semantic nets were used for question answering (qv), automatic programming (qv), and studies of learning (qv), memory, and reasoning (qv). Besides the groups mentioned above, other early implementers include Raphael (26), Reitman (27), Simmons (28), and Shapiro and Woodmansee (29). During the 1970s network systems proliferated and attained a considerable degree of power and sophistication. The most complete reference for that decade is the collection by Findler (30); those systems are also discussed in the remaining sections of this entry. During the 1980s, the boundary lines between network, framelike, and linear forms of logic have been breaking down. Expressive power is no longer a significant argument in favor of network vs. linear forms, since new ideas stated in one notation can usually be adapted to other notations. Instead, the arguments are over secondary issues, such as readability, efficiency, naturalness, theoretical elegance, and ease of typing, editing, and printing.

Besides implementations, theoretical studies that analyze the underlying assumptions have made major contributions to the field. In his classic paper, "What's in a Link," Woods (31) systematically analyzed and criticized the foundations of semantic networks and pointed out weaknesses that subsequent researchers have tried to correct. In a lighter style, but with equally serious intent, Drew McDermott (32) wrote a critique with the unforgettable title "Artificial Intelligence Meets Nat-

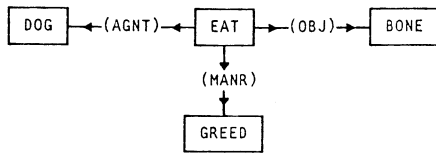


Figure 1. A relational graph for "A dog is greedily eating a bone."

ural Stupidity." Other important analyses were written by Brachman (33), Israel and Brachman (34), Maida and Shapiro (35), Israel (36), and Sowa (4).

### Relational Graphs

The simplest networks used in AI are relational graphs. These graphs consist of nodes connected by arcs. Each node represents a concept, and each arc represents some relationship between the corresponding concepts. Figure 1 shows a relational graph for the sentence "A dog is greedily eating a bone." The four boxes represent concepts of a dog, an act of eating, a bone, and an instance of greed. Labels on the arcs show that the dog is the agent of eating, the bone is the object of eating, and greed is the manner of eating.

The terminology in the field is highly variable. For the sake of this entry, a structure of nodes connected by arcs are called a graph; a structure with nested graphs or higher order relationships between graphs are called a network. But in references to a particular system, the original author's terms are used. Besides terminology, the style of drawing graphs varies from one author to another. Some use ovals instead of boxes; some write labels next to the arcs instead of enclosing them in circles; some abbreviate the labels with single letters like *A* for agent and *O* for object; and others use many different styles of arrows. Figure 2 shows a conceptual-dependency graph in the notation used by Schank (17). The symbol  $\Leftrightarrow$  represents the agent relation. INGEST is one of Schank's primitives: EAT is INGEST with a solid object; DRINK is INGEST with a liquid object; and BREATHE is INGEST with a gaseous object. The extra arrows at the right of the graph show that the bone goes from some unspecified place into the dog.

Since diagrams are hard to type and take a lot of space on the printed page, many authors type their graphs in a more compact notation. For his semantic network processing system [SNePS (qv)], Shapiro (37) developed a linear user language SNePSUL. For his partitioned nets Hendrix (3) developed a linear network language LN2, which was implemented as an extension to INTERLISP. For his conceptual graphs, SOWA (4) uses both a display form like Figure 1 and a linear form like the following:

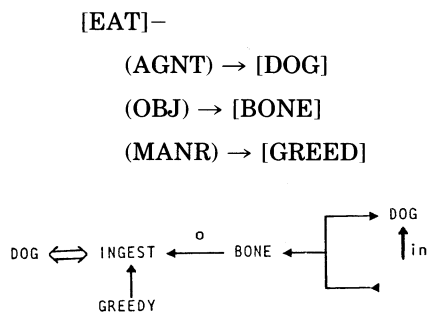


Figure 2. A Schankian form of Figure 1.

In this form square brackets enclose the concept nodes, and parentheses enclose the labels on the arcs. All of the linear forms bear a strong resemblance to frame systems. In fact, Hendrix remarked that LN2 was inspired by the frame system KRL (38).

As a form of logic, relational graphs represent a subset of first-order predicate calculus (see Logic, predicate). A common assumption is that the labels on the nodes represent types, and the presence of a concept node asserts the existence of something of the corresponding type. The arcs correspond to logical relations or predicates. Yet relational graphs have a major limitation: The only logical operators they can easily represent are existence  $\exists$  and conjunction  $\wedge$ . Some versions mark the nodes with other quantifiers, such as  $\forall$ , for all, but the graphs do not clearly show their scope. Despite this limitation, they are powerful enough for many applications.

Commercial database-management systems have similar limitations yet handle enormous volumes of information for businesses and governments around the world.

Simple sentences in natural language can be mapped directly into relational graphs. In particular, they are a good notation for the case frames that are used in linguistics to show the patterns of relations attached to verbs (see Grammar, case).

A kind of relational graph that has become popular for database design is Chen's version of entity-relationship diagrams (39). Instead of quantifiers, ER diagrams mark the arcs to show one-to-one, many-to-one, and many-to-many relationships. Although these diagrams originated in the database field, the boundaries are blurring as databases evolve into knowledge bases.

### Verb-Centered Relational Graphs

Verbs have patterns of case relations that link them to noun phrases. For the sentence "Mary gave a book to Fred," Mary is the agent of "give," the book is the object of "give," and Fred is the recipient of "give." Besides case relations within a sentence, natural languages also express relations between sentences. Such relations are needed for the following kinds of expressions.

**Conjunctions.** The most direct way of putting two sentences together is by joining them with a conjunction. Some conjunctions, like "and," "or," and "if," express logical connectives; others, like "after," "when," "while," "since," and "because," express time and causality.

**Verb complements.** The case frames for many verbs take an embedded sentence, usually as a direct object, but sometimes as some other case. Such verbs include "say," "believe," "think," "know," "persuade," "threaten," "attempt," "try," "help," and "prevent."

**Sentential modifiers.** Many adverbs and prepositional phrases directly modify the verb, but others modify the entire sentence. Such adverbs, like "possibly" or "usually," are often placed at the beginning of a sentence. The phrase "once upon a time" typically modifies an entire story.

**Modes and tenses.** The auxiliary verbs "may," "can," "must," "should," "would," and "could" express modal oper-

ators that govern the entire clause in which they occur. Tenses may be expressed by endings on verbs or by separate adverbs, such as “now,” “later,” “once,” or “tomorrow.”

*Connected discourse.* Besides the relationships expressed within a single sentence, there are more global relationships between sentences of a story, discussion, or explanation. Some of them are not stated explicitly: Time sequence and the steps of an argument are often implied by the order in which sentences are stated.

Since the verb is such an important part of a sentence, many systems make it the centerpiece and link intersentential relations to it. That approach was inspired by the Indo-European languages that express tense and modality with verbal inflections. Consider the example “While a dog was eating a bone, a cat passed by unnoticed.” This sentence states that at some time when the proposition “A dog was eating a bone” was true, the second proposition “A cat passed by unnoticed” was also true. Figure 3 shows a verb-centered graph for that sentence. The conjunction “while” (WHL) links the node PASS-BY to the node EAT. Figure 3 shows that the agent of not noticing is the dog. Although that is the most likely interpretation, the original sentence leaves the agent ambiguous. Figure 3 could be made equally ambiguous by erasing the AGNT relation between the concepts [DOG] and [¬NOTICE].

Verb-centered graphs are relational graphs with the verb considered as the focal point of each proposition. Tense markers like PAST and dyadic relations like WHL are attached directly to the concept nodes that represent verbs. Roger Schank’s conceptual-dependency graphs, e.g., use that approach. Figure 3 could be converted into a Schankian graph by erasing the boxes around the concept nodes and replacing the relations with special arrows, such as  $\Leftrightarrow$  for AGNT. In his earlier work Schank (17) translated all verbs into a small number of primitive acts, such as PTRANS for physical transfer. Concept types like PASS-BY could be eliminated by paraphrasing: The cat started from a point away from the dog, did a PTRANS to a point near the dog, and then did a PTRANS to some point opposite the starting point. In more recent work Schank and Carbonell (40) allow higher level types without requiring them all to be translated into primitives.

In studying human information processing, Norman and Rumelhart (11) did extensive work with verb-centered graphs. They applied them to everything from low-level case relations to high-level patterns of episodes and themes. Yet the psychological evidence was inconclusive: Their networks were consistent with the evidence, but they could not rule out other representations. Although their conclusions were psychologically ambiguous, they did demonstrate the generality and flexibility of the networks as an AI representation.

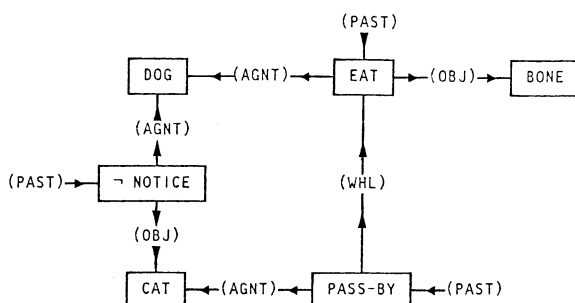


Figure 3. A verb-centered relational graph.

Although verb-centered graphs are highly flexible, they have some serious limitations. One weakness is that they cannot distinguish modifiers that affect only the verb from those that affect the entire sentence. Consider the next two examples:

The dog greedily ate the bone.

Greedily, the dog ate the bone.

In the first sentence the manner of eating was greedy. In the second the situation as a whole was greedy; the manner of eating might actually have been quite refined. Verb-centered graphs cannot distinguish these two cases. They are even more awkward for expressing the nested propositions illustrated in the next section.

Like other forms of relational graphs, verb-centered graphs have difficulty in expressing scope of quantifiers, tenses, and modal operators. Although many researchers have used these graphs for complex problems, they never developed a general method for handling those operators. Figure 3 illustrates the difficulty: The relation PAST is attached to the verb EAT, but not to the bone, which probably ceased to exist after the act of eating. That graph also shows WHL linking PASS-BY to EAT, but it does not show that the cat’s passing by and the dog’s not noticing occurred at the same time. Some notation for showing context is necessary: PAST should modify the entire context of the dog eating the bone, and WHL should link that context to another context where the cat is passing by unnoticed.

### Propositional Networks

In propositional networks certain nodes represent entire propositions. These nodes serve two purposes: they provide points of attachment for intersentential relations; and they define a context that shows the scope of quantifiers and other operators. The next two examples illustrate relations that require proposition nodes:

Sue thinks that Bob believes that a dog is eating a bone.

If a dog is eating a bone, it is unwise to try to take it away from him.

In the first sentence, the verbs “think” and “believe” take entire propositions as their objects: What Bob believes is the proposition “A dog is eating a bone”; what Sue thinks is the more complex proposition “Bob believes that a dog is eating a bone.” Such nesting of propositions within propositions can be iterated any number of times. One way to show the nesting is to introduce proposition nodes that contain nested graphs. Figure 4 shows a propositional network for this sentence. It shows that the object of THINK is a proposition, which contains another proposition as the object of BELIEVE. Note that the experiencer (EXPR) links THINK to Sue and BELIEVE to Bob, but the agent relation (AGNT) links EAT to DOG. The reason for the different relations is that thinking and believing are states that people experience, but eating is an act performed by an agent.

The other example is an implication that relates two propositions. The antecedent is “A dog is eating a bone,” and the consequent is “It is unwise to try to take it away from him.” The infinitives “to try” and “to take” indicate other nested propositions: what is unwise is the proposition “[someone] try

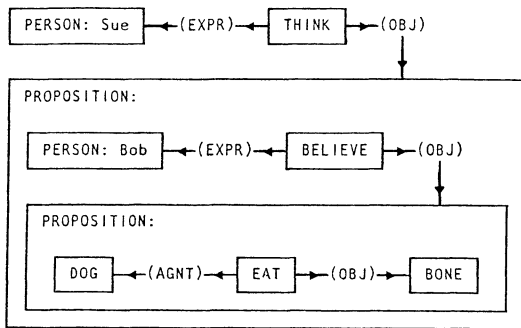


Figure 4. A network with nested propositions.

$P$ ," where  $P$  is another proposition "[someone] take it [the bone] away from him [the dog]." Besides nested propositions, this sentence also requires coreference links to connect the nodes for "it" and "him" to the nodes for the bone and the dog. A coreference link is also needed to show that the person who does the trying is the same as the one who does the taking. Finally, the notation must show the logical quantifiers and their scope. The terms "a dog" and "a bone" indicate existential quantifiers bound within the antecedent of the implication, but their scope includes the consequent. Although there are no explicit quantifiers in the consequent, the intended meaning is that all such instances of trying by any person are unwise (as if the sentence had said "it is always unwise for anyone to try . . ."). Although most notations for semantic networks look similar for simple sentences, they diverge sharply with sentences as complex as this. Figure 5 shows a network for that sentence in the conceptual graph notation developed by Sowa (4).

Figure 5 involves a number of subtle issues. To represent implication, Sowa follows Peirce in translating "if  $p$  then  $q$ " into the form "not ( $p$  and not  $q$ )" or simply  $\neg[p \wedge \neg q]$ . An advantage of this form is that quantifiers in  $p$  (i.e., "a dog" and "a bone") automatically include the pronouns of  $q$  ("him" and "it") in their scope. That property is not true when "if  $p$  then  $q$ " is mapped into the form  $p \supset q$ . The pronouns "him" and "it" are represented by concept nodes of type MALE and ENTITY in

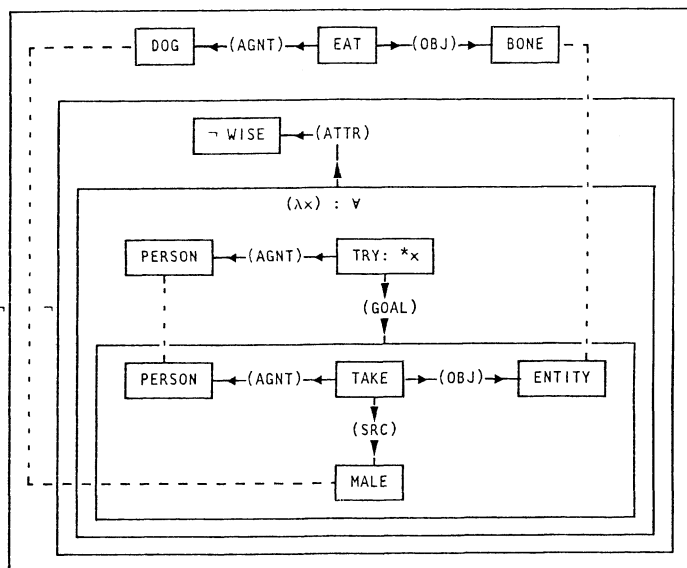


Figure 5. Nested propositions and coreference links.

the innermost context; the dashed lines are the coreference links to the DOG and BONE concepts. The universal quantifier on instances of trying is tricky: All and only those instances of trying that have some person taking the bone away from the dog mentioned in the antecedent have the attribute unwise. Therefore, the universal quantifier  $\forall$  does not range over the type TRY but over a subtype of TRY restricted to agents whose goal is to take the bone away from the dog (the relation SRC indicates source). That restricted type is defined by a lambda abstraction (indicated by  $\lambda x$ ), where the variable  $x$  designates the concept [TRY: \* $x$ ] as the formal parameter. Using lambda expressions to restrict the range of quantification is similar to Altham and Tennant's sortal quantification (41). The type label PROPOSITION, by the way, has been omitted in Figure 5; any box without a type label is assumed to be of type PROPOSITION.

Except for details of drawing concept nodes as boxes or ovals, all relational graphs and verb-centered graphs have strong similarities. Propositional networks, however, differ in several design considerations, shown below.

*Enclosing the context.* Figures 4 and 5 show propositional boxes that enclose the entire context. Some systems, such as SNePS (37), link the components of a proposition to the outside of a proposition node instead of enclosing them inside the node.

*Strict nesting.* Some systems enforce a strict nesting of contexts: the existential graphs by Peirce (1897), the hierarchical graphs by Janas and Schwind (42), and the conceptual graphs by Sowa (4). For his partitioned nets Hendrix (2,3) allowed overlapping contexts: A concept could occur in two different contexts, neither of which was nested in the other.

*Coreference links.* With strictly nested contexts coreference links (the dashed lines in Fig. 5) are necessary to show that two concepts in different contexts refer to the same individual. With overlapping contexts the same concept may occur in both contexts, and coreference links are often unnecessary.

These differences are stylistic conventions that do not affect the logical power. Both Hendrix and Sowa, who enclose their contexts in boxes, implement them with special nodes to which they attach the enclosed nodes. If they had chosen to do so, they could have drawn them without the enclosing boxes. The choice of nested vs. overlapping contexts is also a stylistic choice: strictly nested contexts can always be drawn on a plane; but complex networks, such as Figure 5, can become unreadable or undrawable with overlapping contexts. The need for extra nodes with coreference links between them follows from the constraint on nested contexts. But the extra nodes often reflect pronouns and anaphoric references in natural language; as in Figure 5, the nodes [MALE] and [ENTITY] result from the pronouns "him" and "it."

The first propositional network was Shapiro's MIND system (43), which evolved into SNePS. Shapiro was the first to implement all the operators and quantifiers of first-order logic in a general network notation. As a theorem prover, SNePS is comparable in efficiency to systems that use a linear notation for logic. In fact, one of his students wrote a translator from standard logic to SNePS: input and output were in linear formulas, but the proofs were carried out on the network forms. To demonstrate its use for expert systems, Shapiro (44) used

SNePS to implement a microbiology system called COCCI. Tranchell (45) demonstrated its generality by implementing another network system, KL-ONE (qv), completely in SNePS (qv).

For comparison, Figure 6 shows the SNePS equivalent of Figure 4. The nodes M1, M2, M3, M4, and M5 are proposition nodes: M1 corresponds to the outermost context in Figure 4, which asserts the entire proposition about Sue thinking something; the object of Sue's thought is the proposition M2, which asserts that Bob believes M3; and M3 is the proposition about a dog eating a bone. M2 and M3 correspond to the two boxes of type PROPOSITION in Figure 4. Besides the stylistic convention of enclosing propositions in boxes, these diagrams bring out deeper issues concerning the referents of each node. In Figure 4 the box [PERSONS:Sue] refers to an individual named Sue of type PERSON; the box [DOG] refers to an unnamed individual of type DOG. In Figure 6 the nodes SUE and BOB do not show that those individuals are persons; if relevant, that information would be stated by separate propositions. The nodes B1 and B2 refer to the unnamed dog and bone; the proposition nodes M4 and M5 state that B1 is a member of the DOG class, and B2 is a member of the BONE class. Another difference between these two diagrams is in the treatment of verbs. In Figure 4 the box [THINK] refers to an individual instance of that type. A later sentence might refer back to it: "But her thought was mistaken." Figure 6, however, does not use variable nodes that could support such references. Yet the SNePS system does allow that option: a node for this instance of thinking could be introduced in the same way as the nodes B1 and B2 for the dog and the bone.

Several other researchers have developed highly flexible versions of propositional networks and applied them to a wide variety of AI problems. Schubert (46) and Schubert et al. (47) extended propositional networks to handle essentially every feature that could be expressed in a linear logic. Hendrix (2,3) implemented partitioned nets as a basis for an English query system. Duda et al. (48) used Hendrix's inference engine as a basis for PROSPECTOR, an expert system for mineral exploration. Sowa's conceptual graphs (4,49) have been used in several applications: an English query system with fuzzy referents (50); an expert system for financial auditing (51); and a system for knowledge acquisition from texts (52).

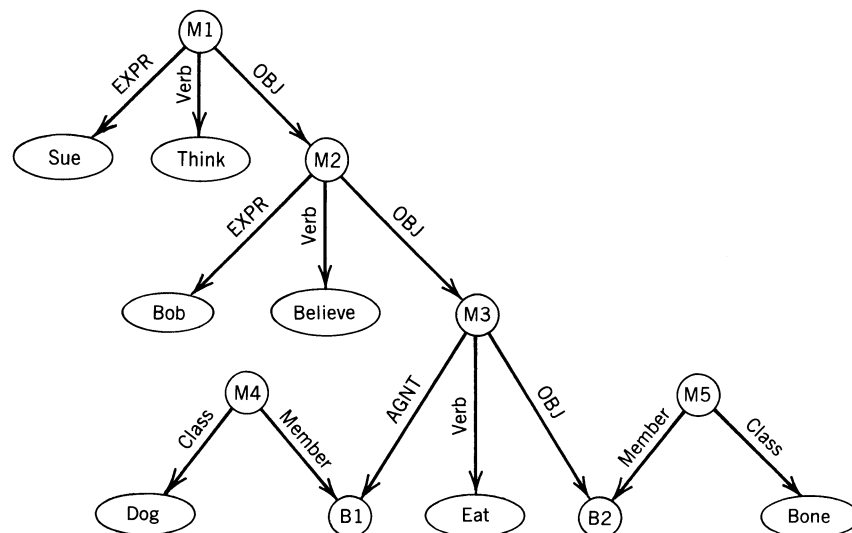


Figure 6. SNePS equivalent of Figure 4.

## Type Hierarchy

A hierarchy of types and subtypes of concepts is a standard feature of most systems of semantic networks. All hierarchies include entities: BEAGLE < DOG < CARNIVORE < ANIMAL < LIVING-THING < PHYS-OBJECT < ENTITY. They may also include events: DONATE < GIVE < ACT < EVENT. And they may include states: ECSTASY < HAPPINESS < EMOTIONAL-STATE < STATE. Aristotle's hierarchy had 10 general categories at the top: substance, quantity, quality, relation, place, time position, state, activity, and passivity. For completeness, some authors introduce a universal type above all the others. Martin (53) called the highest type SUMMUGENUS; other authors use the symbol T, which commonly represents the top of a lattice.

There is no universally accepted terminology for the type hierarchy. They symbol < between a more specialized type and a more general type may be read is a subtype of, is a subset of, is a subpart of, is a kind of, is a flavor of, or simply isa. Unfortunately, there are objections to all these terms.

Bertrand Russell used the word "type" in a more restricted sense in logic—one reason why logicians use the word "sort" in the sense that programmers use the word "type."

The term "isa" is ambiguous, since it could mean either subtype in the sense "A beagle is a dog" or instance in the sense "Snoopy is a beagle."

The terms "set" and "subset" confuse intensions with extensions: the types in a conceptual system are categories of thought, which may or may not correspond to sets of existing entities in the real world.

The term "sort" can cause confusion with the keys used by sorting programs.

The terms "kind" and "flavor" are not widely used.

Of all these terms, "type" and "subtype" are a reasonable and widely accepted compromise. They also emphasize the similarity with abstract data types in programming languages.

The term "hierarchy" is itself confusing. It usually means a partial ordering, where some types are more general than others. The ordering is only partial because many types are not comparable: neither HOUSE < DOG nor DOG < HOUSE.



But note that DOG-HOUSE is a subtype of HOUSE, not of DOG. What is confusing about hierarchies is that many authors do not specify what kind of partial ordering is intended.

*Acyclic graph.* Every partial ordering can be drawn as an acyclic graph—a graph with no cycles. Although an acyclic graph has no cycles, it may have branches that separate and come back together again, permitting some nodes to have more than one parent. Such graphs are sometimes called tangled hierarchies.

*Tree.* The most common hierarchy is a rooted tree. It imposes conditions on an acyclic graph to untangle the hierarchy: there is a single general type at the root of the tree, and every other type  $x$  has exactly one parent  $y$ , which is the minimal supertype of  $x$ .

*Lattice.* Unlike trees, lattices are intended to have nodes with multiple parents. But they impose other constraints: Every pair of types  $x$  and  $y$  must have a minimal common supertype  $x \cup y$  and a maximal common subtype  $x \cap y$ . This constraint causes a lattice to look like a tree from both ends. Instead of a root, a lattice has a maximal node  $\top$  that is a supertype of all others and a minimal node  $\perp$  that is a subtype of all others. (It is also possible to have infinite lattices with no maximal or minimal nodes.)

The first mechanized type lattice was Leibniz's universal characteristic (54). He assigned a prime number to each primitive type and products of primes to each compound type. For any types  $x$  and  $y$ ,  $x$  is a subtype of  $y$  if  $y$  divides  $x$ ; the minimal common supertype  $x \cup y$  is the greatest common divisor of  $x$  and  $y$ ; and the maximal common subtype  $x \cap y$  is the least common multiple. The maximal node  $\top$  is 1, and the minimal node  $\perp$  is the product of all the primitives. One of his motivations for designing the first calculator to do multiplication and division was to automate his system of reasoning. One of the first AI implementations, Masterman's original semantic network (21), was also organized as a lattice.

### Inheritance

A major use for a type hierarchy is to allow properties to be inherited from supertypes to subtypes (see Inheritance hierarchy): whatever is true for any ANIMAL is true for any subtype MAMMAL, FISH, or BIRD. The original inheritance system is Aristotle's theory of syllogisms. According to Lukasiewicz (55), the basic form of an Aristotelian syllogism is a conditional:

If  $A$  is predicated of all  $B$ ,  
and  $B$  is predicated of all  $C$ ,  
then  $A$  is predicated of all  $C$ .

This is the pattern of the first mood, which the medieval Scholastics named Barbara (the three  $a$ 's in "Barbara" indicate three universal affirmative clauses). Aristotle systematically analyzed all combinations of clauses: universal affirmative (A); particular affirmative (I); universal negative (E); and particular negative (O). The syllogism Celarent, e.g., has three clauses of type E, A, E:

If  $A$  is predicated of no  $B$ ,  
and  $B$  is predicated of all  $C$ ,  
then  $A$  is predicated of no  $C$ .

Of the 24 valid moods of syllogisms, the most important and most widely implemented in modern AI systems is Barbara.

With the rise of symbolic logic, the syllogism went into a decline. Logicians tended to ignore it as a variant of quantification theory limited to monadic predicates. Yet the type hierarchy and the mechanism of inheriting properties remain important.

Monadic predicates, represented by common nouns and adjectives, pervade natural languages.

The type hierarchy provides a good structure for indexing a knowledge base and organizing it efficiently.

Following a path through a hierarchy can be much faster than a general theorem prover.

Selectional constraints used in a parsing program primarily depend on inheritance, not on general proof procedures.

Procedures as well as properties can be inherited through a type hierarchy. Such an inheritance method has been implemented in the procedural semantic networks of the TAXIS project (56).

Finally, the syllogism could not have dominated logic for over two millennia if it had not been a vital and natural form of reasoning.

Because of its importance, inheritance methods have been widely used in AI since its earliest days. Two early examples include Masterman's semantic networks (21) and Raphael's SIR (qv) (26).

An important modern system that emphasizes inheritance through the type hierarchy is KL-ONE (qv) (33,57). Figure 7 shows a definition of DOG-HOUSE in a KL-ONE network. The three concepts marked with asterisks (HOUSE, ANIMAL, and DOG) are primitives that have no definition. The circle with a box in it shows a role: the concept HOUSE has a role named "inhabitant." The notation (1,NIL) indicates one or more inhabitants; and the notation  $v/r$  shows that ANIMAL is a value restriction on the kinds of entities that may fill the inhabitant role. The double arrow shows that the newly defined concept DOG-HOUSE is a subtype of HOUSE. The single arrow marked "restricts" shows that DOG further restricts the inhabitant role for the subtype DOG-HOUSE. Subtypes inherit the roles of their supertypes, but they may have more roles and tighter constraints on the roles they inherit.

One problem with inheritance systems is the handling of defaults or partial inheritance (see Reasoning, default). For example, one might associate a general property "can-fly" with the concept type BIRD. Then one could use a canceling operator to block the inheritance of that property by exceptions, such as PENGUIN, OSTRICH, or KIWI. Operators for negating or canceling inheritance have been implemented in many

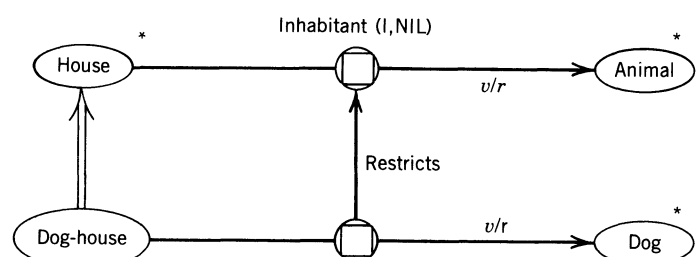


Figure 7. A KL-ONE definition for DOG-HOUSE

systems. But Brachman (58) raised some criticisms: A rock, e.g., could be defined as an elephant except that it does not have a trunk, is not alive, has no feet, etc. With unrestricted use of canceling, an inheritance system turns into a vacuous theory that offers no guidelines for organizing a knowledge base. Newer systems of nonmonotonic logic (see Reasoning, nonmonotonic) are trying to develop systematic ways of handling defaults. Such approaches could also be adapted to a network notation.

### Logic

Since anything that can be expressed in logic can be said in any natural language, a semantic notation must contain all of logic as a subset. But the first implementations of semantic nets had to handle so many complexities of language that logic was not one of the first priorities. After some of the problems of syntax and semantics were solved, network theorists turned to logic. Simmons and Bruce (59) implemented some of the logical operators, but not all. Shapiro (43) had the first system that handled full first-order logic. Hendrix (2) implemented partitioned nets, with a notation that had strong similarities to Peirce's graphs. Schubert (46) implemented modal operators and definite and indefinite descriptions in his graphs. Since then, network systems have been developed into versatile notations that can express anything expressible in any of the linear notations for logic.

One question that network theorists are always asked is "Why bother with networks when standard logic is just as good and much more familiar?" But the common Peano–Russell notation is not "just as good." In fact, as a semantic basis for natural language, it is highly unnatural. For example, the sentence "A cat chased a mouse" might be represented by the formula

$$(\exists x)(\exists y)[\text{cat}(x) \wedge \text{mouse}(y) \wedge \text{chased}(x, y)]$$

This formula uses mathematical machinery of variables, quantifiers, and conjunctions to express something that can be said very simply in any natural language. Even so, it is still inadequate because it fails to represent the past tense of the verb "chased." To represent time, the next formula adds a variable  $t$ :

$$(\exists x)(\exists y)(\exists t)[\text{cat}(x) \wedge \text{mouse}(y) \wedge \text{time}(t) \wedge \text{chase}(x, y, t) \wedge \text{before}(t, \text{now})]$$

Even this notation is inadequate because it shows the time  $t$  as governing only the verb and not the cat or the mouse (which may have ceased to exist after the chase). A more general notation would require the quantifiers that govern  $x$  and  $y$  to be nested inside a context indexed by the time  $t$ :

$$(\exists t)[\text{time}(t) \wedge \text{before}(t, \text{now}) \wedge \text{at}(t, (\exists x)(\exists y)[\text{cat}(x) \wedge \text{mouse}(y) \wedge \text{chase}(x, y)])]$$

This formula is no longer first order since it has a formula nested as an argument of the "at" predicate. It also has an asymmetry between nouns and verbs: the variables  $x$  and  $y$

refer to the cat and the mouse, but the act of chasing has no variable of its own. Such variables would be needed to express anaphoric references in subsequent sentences, such as "The chase lasted 39 seconds." Therefore, the next formula adds another variable  $z$  for the act itself:

$$(\exists t)[\text{time}(t) \wedge \text{before}(t, \text{now}) \wedge \text{at}(t, (\exists x)(\exists y)(\exists z)[\text{cat}(x) \wedge \text{mouse}(y) \wedge \text{chase}(z, x, y)])]$$

But this formula is still asymmetric since the predicates have only one argument for nouns but three for the verb. Furthermore, the relationships between the nouns and verbs are expressed only by the position of the arguments of  $\text{chase}(z, x, y)$ . A more explicit notation would introduce predicates to express case relations:  $\text{agnt}(z, x)$  to show that the agent of chase  $z$  is cat  $x$ , and  $\text{obj}(z, y)$  to show that the object of chase  $z$  is mouse  $y$ .

$$(\exists t)[\text{time}(t) \wedge \text{before}(t, \text{now}) \wedge \text{at}(t, (\exists x)(\exists y)(\exists z)[\text{cat}(x) \wedge \text{mouse}(y) \wedge \text{chase}(z) \wedge \text{agnt}(z, x) \wedge \text{obj}(z, y)])]$$

As this example shows, Peano–Russell notation requires an enormous number of symbols to express common distinctions in ordinary English. In *Principia Mathematica*, Whitehead and Russell developed it as a tool for analyzing the foundations of mathematics. It is well suited to mathematics, but it is awkward as a logical form for natural languages.

Each of the network systems has a different notation. But as an example, Figure 8 shows a conceptual graph for the sentence "A cat chased a mouse." Every concept implicitly asserts the existence of something of the corresponding type. This graph asserts the existence of a cat, an instance of chasing, and a mouse. It further asserts that the cat is the agent of chasing and the mouse is the object of chasing. To represent the past occurrence of this situation, the monadic relation PAST is attached to a context (a proposition node) that encloses the entire graph. To save space on the printed page, Figure 8 can also be written in the linear form with square brackets to represent concepts and parentheses to represent conceptual relations:

$$(\text{PAST}) \rightarrow [[\text{CAT}] \leftarrow (\text{AGNT}) \leftarrow (\text{CHASE}) \rightarrow (\text{OBJ}) \rightarrow [\text{MOUSE}]]$$

The relation PAST is not a primitive. In the next section it is defined in terms of other relations PTIM (point in time) and SUCC (successor). Whether the relation PAST or its definition is used, the conceptual graph is shorter, simpler, and more readable than the Peano–Russell form. Yet it expresses every distinction expressed in the Peano–Russell form.

To represent operators other than conjunction and existence, Peirce used contexts to show the scope of negations and modality. Then disjunctions, universal quantifiers, and other operators could be introduced by definitions. Figure 5 shows a graph for the sentence "If a dog is eating a bone, then it is unwise to try to take it away from him." That graph contains a universal quantifier that ranges over a type defined by  $\lambda$ -abstraction. By expanding those operators and simplifying the result, Figure 9 may be derived. The new graph represents the sentence "If a dog is eating a bone and a person is trying to take it away from him, then that try is unwise." This new sentence is implied by the original; rules of inference demonstrate the implication by transforming Figure 5 into Figure 9.

Readability is a major reason for preferring graphs to a linear form of logic. Anyone who doubts that claim should try expressing Figure 9 in some variant of Peano–Russell nota-

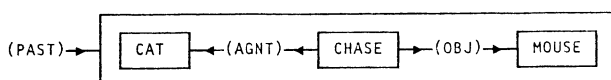


Figure 8. A conceptual graph for "A cat chased a mouse."

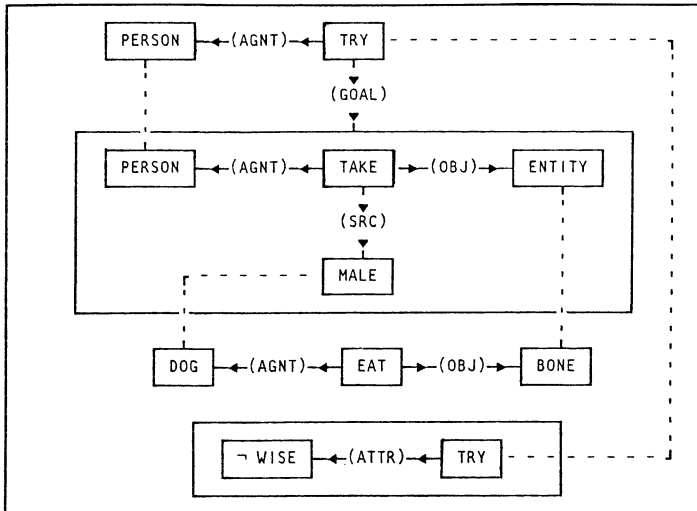


Figure 9. Result of expanding the  $\forall$  and  $\lambda$  operators in Figure 5.

tion. Graphs also have an important computational advantage: they can keep all the information about an entity at a single node and show related information by arcs connected directly to that node. Systems based on graphs take advantage of that connectivity in algorithms for scanning the graphs to generate language or perform inferences. By contrast, linear notations tend to scatter related pieces of information all through a formula or collection of formulas. Montague grammar, which uses a linear logic to represent the semantics of natural language, is notoriously complex. Much of that complexity, however, results from the need to deal with variables in a linear notation. The equivalent operations on graphs are much simpler.

### Taxonomic and Assertional Systems

The subtype links in a semantic network are of a totally different nature from the arcs of a relational graph. In Figure 1, e.g., the AGNT relation links some instance of DOG to some instance of EAT. It makes a simple, first-order statement about two individual instances. A subtype link from DOG to CARNIVORE, however, is a higher order relation between types: It implies that every dog is a carnivore, not just some instance denoted by a particular concept node. The difference in nature between these kinds of arcs has led to a great deal of confusion in the AI literature. Depending on which kinds of links they emphasize, semantic network systems have diverged in two directions.

Assertional systems emphasize the ability to make statements, but they often neglect the type hierarchy. Schank's conceptual-dependency graphs (17), e.g., were designed to represent the meaning of sentences in natural languages. Yet Schank has not emphasized the type hierarchy, and he even denied that his system should be called a semantic net.

Taxonomic systems emphasize the type hierarchy and methods for defining new types, but they tend to neglect the propositional aspects. Masterman's original semantic networks (21), Quillian's semantic memory (9), Martin's OWL (53), and Brachman's KL-ONE (qv) (33) emphasized definitional mechanisms, but they were all weak in assertional propositions.

The divergence between assertional and taxonomic systems began with the earliest systems of logic. Aristotle was a taxonomist with his method of defining new types by "genus" and "differentiae." Each new type or species is defined by stating its genus or supertype and the properties or differentiae that distinguish it from other subtypes of the same genus. Aristotle's method has formed the basis for all dictionaries written in the past two millennia.

Symbolic logic takes the opposite extreme of emphasizing assertions while ignoring the taxonomy of terms. For example, it uses terms like  $R(x, y)$  for showing some relation  $R$  between  $x$  and  $y$ . Instead of giving an explicit definition for  $R$ , logic provides ways of asserting axioms about  $R$ . If the axioms are detailed enough,  $R$  may be completely, though indirectly, determined. But with a less detailed set of axioms,  $R$  is open to many possible interpretations. In that case  $R$  is just a syntactic marker with little or no meaning.

By means of  $\lambda$ -abstractions, a definitional structure can be built on top of an assertional system. A certain number of types may be taken as primitive, and new ones can then be introduced by definition. The following graph, e.g., asserts the proposition "A dog lives in a house" (STAT is the state relation, and LOG is the location relation):

$$[\text{DOG}] \rightarrow (\text{STAT}) \rightarrow [\text{LIVE}] \rightarrow (\text{LOC}) \rightarrow [\text{HOUSE}]$$

By parameterizing this proposition, one can define predicates for new types. The symbol  $(\lambda x)$  is used to designate some concept node flagged by  $*x$  as the formal parameter. By designating the DOG node as the parameter, the following  $\lambda$ -abstraction defines a new type HOUSE-DOG as a subtype of DOG that lives in a house:

$$\text{HOUSE-DOG} = (\lambda x) [\text{DOG}: *x] \rightarrow$$

$$(\text{STAT}) \rightarrow [\text{LIVE}] \rightarrow (\text{LOC}) \rightarrow [\text{HOUSE}]$$

By designating HOUSE as the parameter, the next  $\lambda$ -abstraction defines DOG-HOUSE as a type of HOUSE where a dog lives:

$$\text{DOG-HOUSE} = (\lambda x) [\text{DOG}] \rightarrow$$

$$(\text{STAT}) \rightarrow [\text{LIVE}] \rightarrow (\text{LOC}) \rightarrow [\text{HOUSE}: *x]$$

New relations can also be introduced by  $\lambda$ -abstractions. The past-tense relation, e.g., marks a PROPOSITION whose point in time (PTIM) is a time that has a successor (SUCC), which is the privileged time Now

$$\text{PAST} = (\lambda x) [\text{PROPOSITION}: *x] \rightarrow$$

$$(\text{PTIM}) \rightarrow [\text{TIME}] \rightarrow (\text{SUCC}) \rightarrow [\text{TIME}: \text{Now}]$$

A  $\lambda$ -abstraction can also be placed in the type field of a concept node. In Figure 5, e.g., the  $\lambda$ -abstraction defines a subtype of TRY whose agent has the goal of taking the bone away from the dog. The quantifier  $\forall$  ranges over that special subtype of TRY. Schubert (46) first introduced  $\lambda$ -abstractions into semantic networks. Sowa (4,60) elaborated them into a versatile definitional system. Although Shapiro (37) does not use the  $\lambda$  symbol, his pattern nodes in SNePS perform a similar function in definitions.

Since the definitional component of KL-ONE is strong but its assertional component is weak, two different hybrid systems evolved by adding an assertional language to it: KRYPT-

TON (61,62) and KL-TWO (63). In each the KL-ONE component became the taxonomic language for defining concept types. A version of first-order predicate calculus became the assertional languages for using the types. The hybrid systems, however, suffer from a curious anomaly:

the taxonomic component supports a kind of modality for expressing optional and obligatory features of a concept,  
but the assertional component is a first-order language that cannot assert modal or intensional propositions.

Figures 4, 5, and 9 have propositions nested as objects of intensional verbs. Such assertions cannot be stated in first-order logic. The definitional component of KL-ONE also has limitations. With  $\lambda$ -abstractions, e.g., one could define a special type of FARMER with either red or blond hair who owns a donkey that he possibly beats on those Fridays when he is not drunk. KL-ONE could not express that definition since it requires complex assertions.

Although the hybrid systems are not as general or expressive as  $\lambda$ -abstractions on a propositional base, they can be quite efficient for the cases they handle. The highly structured type hierarchy improves efficiency by allowing properties to be inherited by subtypes without requiring separate proofs for each type. In effect, the hybrid systems combine an Aristotelian style of syllogistic with a resolution theorem prover, but similar techniques can also be applied to type hierarchies defined by  $\lambda$ -abstractions.

### Generic and Individual Concepts

In mapping language to logic, common nouns like "cat" map into monadic predicates like  $CAT(x)$ . But proper names like "Felix" map into individual constants. The term "isa" tends to blur this distinction: one says "Felix is a cat" in the same form as "A cat is an animal." Consequently, many AI systems label their nodes with "cat" or "Felix" in the same format. The first network implementation, Ceccato's correlation nets (24), had distinct relations for the two, but many later systems confused them. The confusion is compounded by terminology that differs from one author to another. Several different notions must be distinguished.

*Particular individuals.* In logic constants designate individuals in the real world or some possible world. The fictional cat Felix, e.g., does not exist in the real world, but it is a well-defined character in a former comic strip. Many systems introduce special nodes or notations for individuals. Sowa (4), e.g., writes a name or identifier after the type in a concept node:  $[CAT: Felix]$  represents an individual concept of a cat named Felix.

*Indefinite references.* The English phrase "a cat" refers to some individual of type CAT but does not say which one. Hilbert introduced his "epsilon operator" to express such terms: the notation  $\epsilon xCAT(x)$  represents an arbitrary  $x$  for which the predicate  $CAT(x)$  is true (64). Sowa uses the notation  $[CAT]$ , called a generic concept, to represent an arbitrary individual of type CAT. Its referent is the epsilon term  $\epsilon xCAT(x)$ .

*Anaphoric references.* After an individual has been mentioned, a later pronoun or noun phrase may refer back to it. In semantic nets it is common to join all the nodes in a

context that refer to the same individual. Yet when a parser finds an anaphoric reference, such as "the cat," it may not be clear which cat is meant. The parser may therefore flag the concept node with a marker such as # to indicate a reference to be resolved later: The concept  $[CAT: \#]$  would later be linked to some other cat node in the current context or an enclosing context.

*Definite descriptions.* Besides being used for anaphoric references, the definite article "the" is also used to introduce a unique individual that has given property. For example, "the fourteenth president of the United States" denotes a unique person, but most people would not be able to identify that person without checking a history book. Definite descriptions can be handled as special cases of anaphoric references: If a referent cannot be found, a new individual concept may be introduced into the context and marked with the specified properties.

*Types.* The nodes that represent types should be distinguished from the nodes that represent individuals. In relating semantic networks to logic, Sowa maps concept types to predicates; referents of concepts to individual expressions; and concepts and conceptual graphs to formulas. For the generic concept  $[CAT]$ , the type predicate is  $CAT(x)$ ; the referent is the epsilon term  $\epsilon xCAT(x)$ ; and the formula is the result of applying the predicate to the referent,  $CAT(\epsilon xCAT(x))$ , which is equivalent to the formula  $\exists xCAT(x)$ .

Not all versions of semantic networks distinguish these five different notions. Schubert (46) was the first to introduce definite and indefinite descriptions into his version of semantic networks. Brachman (33) emphasized the distinction between generic and individual concepts. But he did not have a general method for handling definite, indefinite, and anaphoric references.

### Intensions and Extensions

To say that a semantic network represents semantics is circular: A graph notation is just another kind of syntax. To give it some semantic content, there must be an independent basis for determining the meaning of its nodes and arcs. In talking about meaning, philosophers have drawn a distinction between intension of a word (its basic meaning in itself) and its extension (the set of things it refers to). Frege (1) gave the example of "evening star" vs. "morning star." These two terms have different intensions: one means a star that is seen in the morning, and the other means a star that is seen in the evening. Yet both of them have the same extension, namely the planet Venus. In defining a semantic basis, one must decide whether it is extensional (a definition that lists the individuals a term refers to) or intensional (a definition by properties or criteria without any concern for the existence of things that have those properties).

In logic first-order predicate calculus is purely extensional. For a given model the meaning of any predicate is completely determined by listing the set of individuals for which it is true. Modal logic, however, is intensional: It deals with possibilities that do not exist and may never exist. Its model theory requires infinite families of infinite worlds together with an accessibility relation that shows how the worlds are related (65). In such a system extensional definitions are impossible because there is no way to list an infinite number of possibilities.

Most knowledge-representation languages (linear as well as network) are designed to specify intensions. Taxonomic systems define concept types without regard for the existence of the corresponding individuals: The nonexistence of unicorns is irrelevant to the definition of UNICORN as a subtype of MAMMAL. Janas and Schwind (42) gave explicit rules for relating intensions to extensions. Their networks are intensional; they specify the meaning of a sentence independent of any interpretation. But they also defined a method of evaluating the extensions: Given sets of individuals and relations between individuals, the extension of a network is determined by projections into the sets.

Sowa (4) used conceptual graphs to represent both extensional models of the world and intensional propositions about the world (see also Ref. 35). To determine the truth or falsity of a proposition, he defined projection rules similar to Janas and Schwind's to map the propositional graphs into the model graphs. If the model is a traditional closed-world model, the resulting semantics is equivalent to first-order model theory. If the model leaves many properties unknown or unstated, the result is a version of open-world semantics with the truth of many propositions left unknown. If the model has families of contexts that represent different possibilities, different versions of modal semantics can be derived.

### Language Parsing and Generation

Semantic nets can aid a parser in resolving syntactic ambiguities (see Parsing). Without such a representation the burden of language analysis rests on syntactic rules supplemented with tests of semantic features. The structure of a semantic net, however, shows the normal ways that concepts link together. When the parser finds an ambiguity, it can use the net as a guide in selecting one option or another. Several different parsing techniques have been used with them.

*Syntax-directed parsing.* The parser is controlled by a phrase-structure grammar augmented with structure-building and testing operators. As the input is analyzed, the structure-building operators construct a semantic net, and the testing operators check constraints on the partially built net. If the constraints are not met, the current grammar rule is rejected, and the parser tries another option. This approach is one of the most common.

*Semantic parsing.* A semantic parser runs like a syntax-directed parser, but its categories are high-level concept types like SHIP and TRANSPORT instead of syntactic categories like Noun-Phrase or Verb-Phrase. The LADDER system (3) was one of the best known systems that used a semantic parser to generate a semantic net (see Grammar, semantic).

*Conceptual parsing.* The semantic network itself shows expected constraints on the way words may be related and leads to expectations for other words that may occur in the sentence. The verb "give," e.g., requires an animate subject and raises expectations for a recipient and an object given. Schank (17) has been one of the strongest advocates of conceptual parsing (see Parsing, expectation-driven).

*Word-expert parsing.* Because of all the irregularities in natural languages, some people have abandoned the search for universal generalizations and have implemented their dictionaries as collections of independent procedures called

word experts (see Parsing, word-expert). The analysis of a sentence is treated as a cooperative process between word experts, each of which may be as arbitrary and irregular as necessary to handle special cases and exceptions. The chief proponent of this approach has been Small (66).

Arguments over parsing techniques are usually debated with more religious fervor than hard facts. One of the few projects that has experimented with all these techniques is the Semantic Representation Language (SRL) Project at the Technical University of Berlin (67,68). Over the years they have written four different parsers for analyzing German and mapping it into the SRL network form.

The first parser was a Schankian-style conceptual parser. Adding new words to its lexicon was easy, but it could only parse simple sentences and relative clauses. Broadening its syntactic coverage was difficult.

Their second parser was a semantic-oriented augmented transition net (see Grammar, augmented-transition-network). Making the syntax more general was easy, but it ran slower than the first parser.

Next, they implemented a word-expert parser. Handling special cases was easy, but the distribution of the grammar among many separate procedures made it difficult to understand, modify, and maintain.

The most recent parser is a syntax-directed one based on a generalized phrase-structure grammar (qv) (GPSG). It is the most general and systematic, and it is also reasonably fast.

These results are similar to the experiences of many other computational linguists—syntax-directed parsers are usually the most general, but a good set of network operators is necessary for a smooth interface between the grammar and the semantic net.

Generating language from a semantic network is the inverse of parsing: Instead of parsing a linear string to generate a network, the language generator parses a network to generate a linear string. In fact, Shapiro (69) developed a highly symmetric version of an ATN that used exactly the same kinds of rules to generate or to parse. In one of the early systems Quillian (9) generated English from a network simply by tracing a path, mapping concepts along the path into words and relations into prepositions and other syntactic markers. Although Quillian's method was highly restricted, McNeill (70) and Sowa (71) elaborated it into a more general theory of the utterance path. Instead of focusing on the utterance path, syntax-directed approaches control the generation by grammar rules that use the network to determine which rule to apply next. Yet in practice, both methods have an underlying similarity: the utterance path may be regarded as the sequence of nodes that are processed by a syntax-directed generator.

### Machine Learning

Graphs and networks are good notations for programs that try to learn new structures. What makes them good is the ease of adding, deleting, and comparing nodes and arcs. Such operations are usually easier to perform on graphs than on linear notations. Following are some learning programs that used versions of semantic networks:

Winston (72) used a form of relational graphs to describe structures, such as arches and towers. When given positive and negative examples of various structures, his program would construct graphs that showed the patterns of necessary and sufficient conditions for each type of structure.

Salveter (73) used verb-centered relational graphs to represent the case relations associated with various verbs. For each type of verb, her program MORAN would infer the expected case relations by comparing descriptions of scenes before and after the actions described by those verbs.

Schank (74) developed his theory of MOPs [memory-organization packets (qv)] to explain how people learned general information from particular experiences. A MOP is a highly general graph structure that contains the essential relationships abstracted from many specific graphs that describe particular experiences.

Haas and Hendrix (75) developed the NANOKLAUS system that would learn type hierarchies by being told. Instead of requiring a person to give detailed definitions of each concept type, NANOKLAUS would carry on a dialogue to determine the necessary and sufficient conditions.

### Procedural Attachments

Semantic networks are a declarative form of knowledge representation. In that respect, they resemble logic and frame systems. Yet many declarative systems provide an escape mechanism for attaching procedures, either as a set of built-in primitives or as some form of subroutine calls to programs written in another language. Such procedural attachments serve several purposes.

*Efficiency.* When a well-defined algorithm exists, compiled code is usually faster than a more general AI system. An extreme case is a call to built-in arithmetic operators instead of a theorem prover that reduces everything to Peano's axioms.

*External interfaces.* An AI program may be one component of a larger system. It may have to interact with programs in other languages, external databases, the operating system, or a network of other computers.

*Unsolved problems.* A purely declarative system requires an axiom for every aspect of a problem. When a problem is not completely understood, it may be possible to write a "quick and dirty" procedure that handles an important aspect of it.

The TAXIS project at the University of Toronto has concentrated on procedural semantic networks (PSNs) that emphasize programming aspects (56). Unlike propositional networks, which are based on logic, PSNs have an operationally defined semantics. They are especially useful for modeling systems that are changing and evolving through time. PSNs do have some declarative aspects: They support a type hierarchy that allows procedures to be inherited from supertypes to subtypes.

Hybrid approaches that combine procedural and declarative networks have also been designed. Roussopoulos (76) had two interconnected networks: a propositional network that was purely declarative and a network of procedural nodes that supported computation. Concept nodes (but not relation nodes) could be shared by both networks. Sowa (4,49) developed a similar system of dataflow graphs bound to conceptual graphs;

he called the computational nodes actors to emphasize the parallels with other AI systems (see also Refs. 44 and 77).

### Implementation Level

Semantic networks can be implemented in almost any programming language on any machine. LISP and PROLOG are the most popular languages, but versions have been implemented in FORTRAN, SNOBOL, PL/I, Pascal, APL, C, and various assembly languages. A large memory is important for storing all the nodes and arcs, but the pioneering systems of the 1960s were implemented on machines that were smaller and slower than today's personal computers.

One language that is specifically designed for mapping natural languages to and from a network form is Heidorn's PLNLP (Programming Language for Natural-Language Processing). An earlier version, called NLP, was used to implement verb-centered relational graphs for an automatic programming system (78,79). The PLNLP version has been used to implement a large grammar with a very broad coverage of English (80). Sowa and Way (81) used PLNLP to implement a semantic interpreter that generated conceptual graphs from Jensen and Heidorn's parse trees. PLNLP supports two basic kinds of rules: decoding rules parse a linear language and build a network; encoding rules scan a network and generate either linear strings or a transformed network. The semantic interpreter, e.g., used encoding rules to map parse trees into conceptual graphs. Following is an example of an encoding rule for generating a sentence S consisting of a noun phrase NP followed by a verb phrase VP:

$$S \rightarrow NP(\%AGNT(S)) VP(\%S, NUMB \\ = NUMB (AGNT), -AGNT)$$

The basic form is a phrase-structure rule ( $S \rightarrow NP VP$ ) augmented with structure-building and testing operations in parentheses. If the current node is S, this rule constructs a new node of type NP, which is a copy (indicated by the symbol %) of the node linked via AGNT to the S node; then it constructs a node of type VP, which is copy of the S node (indicated by %S), but with the NUMB attribute (indicating singular or plural) set to the NUMB attribute of the node linked via AGNT, and with the AGNT link deleted (-AGENT). Following is a decoding rule for parsing a verb phrase with a direct object:

$$VP(TRANS, \neg OBJ) NP \rightarrow VP(OBJ = NP)$$

This rule says that if a verb phrase VP has a transitive attribute (TRANS) and no object (OBJ) and it is followed by a noun phrase NP, then construct a VP node, which is the same as the previous VP node but with its OBJ set to point to the NP node. As these two examples show, PLNLP rules concisely express the complex operations needed for processing graphs.

Besides languages for semantic networks, special hardware has also been designed. With conventional computers the operations of parsing languages and scanning networks can be implemented quite efficiently. For large knowledge bases, however, the problem of finding the necessary rules or background knowledge may be time consuming. To allow multiple searches to proceed in parallel, Fahlman (82) designed his NETL system as a semantic network that could be implemented with parallel hardware. The intent is to model associative searches in the human brain by passing markers that could follow different paths simultaneously. Waltz and Pollack



(83) designed parallel hardware to search for the most likely interpretation of ambiguous phrases in natural language. Their approach is similar to Quillian's spreading activations (9), which propagate very slowly on a serial machine but which could be executed quickly with parallel hardware.

## BIBLIOGRAPHY

1. G. Frege, *Begriffsschrift*, in J. Van Heijenoort (ed.), *From Frege to Gödel, 1879–1931*, Harvard University Press, Cambridge, MA, pp. 1–82, 1967.
2. G. G. Hendrix, Expanding the Utility of Semantic Networks through Partitioning, *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, pp. 115–121, 1975.
3. G. G. Hendrix, Encoding Knowledge in Partitioned Networks, in Ref. 30, pp. 51–92.
4. J. F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, Reading, MA, 1984.
5. D. D. Roberts, *The Existential Graphs of Charles S. Peirce*, Mouton, The Hague, 1973.
6. O. Selz, *Über die Gesetze des Geordneten Denkverlaufs*, Spemann, Stuttgart, 1913.
7. O. Selz, *Zur Psychologie des Produktiven Denkens und des Irrtums*, Friedrich Cohen, Bonn, 1922.
8. A. D. de Groot, *Thought and Choice in Chess*, Mouton, The Hague, 1965.
9. M. R. Quillian, Semantic Memory, Report AD-641671, Clearinghouse for Federal Scientific and Technical Information, abridged version in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, pp. 227–270, 1968.
10. N. H. Frijda and A. D. de Groot, *Otto Selz: His Contributions to Psychology*, Mouton, The Hague, 1981.
11. D. A. Norman, D. E. Rumelhart, and the LNR Research Group, *Explorations in Cognition*, Freeman, San Francisco, 1975.
12. J. R. Anderson and G. H. Bower, *Human Associative Memory: A Brief Edition*, Erlbaum Associates, Hillsdale, NJ, 1980.
13. D. G. Hays, On the Value of Dependency Connections, *Proceedings of the 1961 International Conference on Machine Translation*, pp. 577–591, 1961.
14. D. G. Hays, "Dependency theory: A formalism and some observations," *Language* 40(4), 511–525 (1964).
15. S. Klein and R. F. Simmons, "Syntactic dependence and the computer generation of coherent discourse," *Mechan. Transl.* 7 (1963).
16. R. C. Schank and L. G. Tesler, A Conceptual Parser for Natural Language, *Proceedings of the First International Joint Conference on Artificial Intelligence*, Washington, DC, pp. 569–578, 1969.
17. R. C. Schank (ed.), *Conceptual Information Processing*, North-Holland, Amsterdam, 1975.
18. D. J. Allerton, *Valency and the English Verb*, Academic, New York, 1982.
19. B. Vauquois and C. Boitet, "Automated translation at Grenoble University," *Computat. Ling.* 11(1), 28–36 (1985).
20. L. Tesnière, *Éléments de Syntaxe Structurale*, 2nd ed., Librairie C. Klincksieck, Paris, 1965.
21. M. Masterman, Semantic Message Detection for Machine Translation, using an Interlingua, *Proceedings of the 1961 International Conference on Machine Translation*, pp. 438–475, 1961.
22. Y. A. Wilks, *Grammar, Meaning, and the Machine Analysis of Language*, Routledge & Kegan Paul, London, 1972.
23. Y. A. Wilks, "An intelligent analyzer and understander of English," *CACM* 18(5), 264–274 (1975).
24. S. Ceccato, *Linguistic Analysis and Programming for Mechanical Translation*, Gordon and Breach, New York, 1961.
25. S. Ceccato, "Automatic translation of languages," *Inf. Stor. Retr.* 2(3), 105–158, 194.
26. B. Raphael, SIR: A Computer Program for Semantic Information Retrieval, Ph.D. Dissertation, MIT, abridged version in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, pp. 33–145, 1968.
27. W. R. Reitman, *Cognition and Thought*, Wiley, New York, 1965.
28. R. F. Simmons, "Storage and retrieval of aspects of meaning in directed graph structures," *CACM* 9, 211–215 (1966).
29. S. C. Shapiro and G. H. Woodmansee, A Net Structure Based Relational Question Answerer, *Proceedings of the First International Joint Conference on Artificial Intelligence*, Washington, DC, pp. 325–346, 1969.
30. N. V. Findler (ed.) *Associative Networks: Representation and Use of Knowledge by Computers*, Academic, New York, 1979.
31. W. A. Woods, What's in a Link: Foundations for Semantic Networks, in Bobrow and Collins (eds.), pp. 35–82, 1975.
32. D. V. McDermott, "Artificial intelligence meets natural stupidity," *SIGART Newsl.* (57) (April 1976); reprinted in J. Haugeland (ed.), *Mind Design*, MIT Press, Cambridge, MA, pp. 143–160, 1976.
33. R. J. Brachman, On the Epistemological Status of Semantic Networks, in Ref. 30, pp. 3–50.
34. D. J. Israel and R. J. Brachman, Distinctions and Confusions, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC, pp. 452–458, 1981.
35. A. S. Maida and S. C. Shapiro, "Intensional concepts in propositional semantic networks," *Cog. Sci.* 6(4), 291–330 (1982). Reprinted in R. J. Brachman and H. J. Levesque (eds.), *Readings in Knowledge Representation*, Morgan-Kaufman, Los Altos, CA, pp. 170–189, 1985.
36. D. J. Israel, "Interpreting network formalisms," *Comput. Math. Appl.* 9(1), 1–13 (1983).
37. S. C. Shapiro, The SNePS Semantic Network Processing System, in Ref. 30, pp. 179–203.
38. D. G. and T. Winograd, "An overview of KRL, a knowledge representation language," *Cog. Sci.* 1, 3–46 (1977).
39. P. P.-S. Chen, "The entity-relationship model—toward a unified view of data," *ACM Trans. Datab. Sys.* 1(1), 9–36 (1976).
40. R. C. Schank and J. G. Carbonell, Jr., Re: The Gettysburg Address, Ref. 30, pp. 327–362.
41. J. E. J. Altham and N. W. Tennant, Sortal Quantification, in E. L. Keenan (ed.), *Formal Semantics of Natural Language*, Cambridge University Press, New York, pp. 46–58, 1975.
42. J. M. Janas and C. B. Schwind, Extensional Semantic Networks, in Ref. 30, pp. 267–302.
43. S. C. Shapiro, A Net Structure for Semantic Information Storage, Deduction and Retrieval, *Proceedings of the Second International Joint Conference on Artificial Intelligence*, London, pp. 512–523, 1971.
44. S. C. Shapiro, COCCI: A Deductive Semantic Network Program for Solving Microbiology Unknowns, Technical Report 173, Department of Computer Science, SUNY at Buffalo, 1981.
45. L. M. Tranchell, A SNePS Implementation of KL-ONE, Technical Report 198, Department of Computer Science, SUNY at Buffalo, 1982.
46. L. K. Schubert, Extending the Expressive Power of Semantic Networks, *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, pp. 158–164, 1975.
47. L. K. Schubert, R. G. Goebel, and N. J. Cercone, in Ref. 30, pp. 121–175.
48. R. J. Duda, Gaschnig and P. Hart, Model Design in the PROSPECTOR Consultant System for Mineral Exploration, in D. Michie

- (ed.), *Expert Systems in the Micro-Electronic Age*, Edinburgh University Press, Edinburgh, pp. 153–167, 1979.
49. J. F. Sowa, "Conceptual graphs for a database interface," *IBM J. Res. Devel.* **20**(4), 336–357 (1976).
  50. S. K. Morton and J. F. Baldwin, Conceptual Graphs and Fuzzy Qualifiers in Natural Language Interfaces, presented at the Cambridge Conference on Fuzzy Sets, 1985. To appear in *Fuzzy Sets and Systems*, 1986.
  51. B. J. Garner and E. Tsui, "Knowledge representation for an audit office," *Austral. Comput. J.* **17**(3), 106–112 (1985).
  52. J. Fargues, M. C. Landau, A. Dugourd, and L. Catach, "Conceptual graphs for semantics and knowledge processing," *IBM J. Res. Devel.* **30**(1) (1986).
  53. W. A. Martin, "Descriptions and the Specialization of Concepts," P. H. Winston and P. H. Brown (eds.), *Artificial Intelligence: an MIT Perspective*, MIT Press, Cambridge, MA, pp. 375–419, 1979.
  54. G. W. Leibniz, *Elementa Characteristica Universalis*, in L. Couturat (ed.), *Opusculum et Fragments inédits de Leibniz*, Ancienne Librairie Germer Baillière, Paris, pp. 42–92, 1903.
  55. J. Lukasiewicz, *Aristotle's Syllogistic from the Standpoint of Modern Formal Logic*, Clarendon, Oxford, 1957.
  56. H. Levesque and J. Mylopoulos, "A Procedural Semantics for Semantic Networks," in Ref. 30, pp. 93–120.
  57. R. J. Brachman and J. G. Schmolze, "An overview of the KL-ONE knowledge representation system," *Cog. Sci.* **9**(2), 171–216 (1985).
  58. R. J. Brachman, "I lied about the trees' or, defaults and definitions in knowledge representation," *AI Mag.* **6**(3), 80–93 (1985).
  59. R. F. Simmons and B. F. Bruce, "Some relations between predicate calculus and semantic network representations of discourse," *Proc. of the Second IJCAI*, London, 1971, pp. 524–530.
  60. J. F. Sowa, Definitional Mechanisms for Conceptual Graphs, in V. Claus, H. Ehrig, and G. Rozenberg (eds.), *Graph Grammars and Their Application to Computer Science and Biology*, Springer-Verlag, Berlin, pp. 426–439, 1979.
  61. R. J. Brachman, R. E. Fikes, and H. J. Levesque, "Krypton: A functional approach to knowledge representation," *Computer* **16**(10), 67–73 (1983).
  62. R. J. Brachman, V. P. Gilbert, and H. J. Levesque, An Essential Hybrid Reasoning System, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 532–539 1985.
  63. M. Vilain, An approach to hybrid knowledge representation, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985.
  64. A. C. Leisenring, *Mathematical Logic and Hilbert's  $\epsilon$ -Symbol*, Gordon and Breach, New York, 1969.
  65. S. A. Kripke, "Semantical analysis of modal logic I," *Zeitschr. Math. Log. Grundle. Math.* **9**, 67–96 (1963).
  66. S. Small, Word Expert Parsing, Technical Report 954, Department of Computer Science, University of Maryland, 1980.
  67. C. Habel, C. R. Rollinger, A. Schmidt, and H. J. Schneider, A Logic Oriented Approach to Automatic Text Understanding, in L. Bolc, (ed.), *Natural Language Based Computer Systems*, Hanser/Macmillan, 1980.
  68. C. R. Rollinger (ed.), *Probleme des (Text-)Verstehens, Ansätze der Künstlichen Intelligenz*, Max Niemeyer Verlag, Tübingen, 1984.
  69. S. C. Shapiro, "Generalized augmented transition network grammars for generation from semantic networks," *Am. J. Computat. Ling.* **8**(1), 12–25 (1982).
  70. D. McNeill, *The Conceptual Basis of Language*, Erlbaum, Hillsdale, NJ, 1979.
  71. J. F. Sowa, "Generating language from conceptual graphs," *Comput. Math. Appl.* **9**(1), 29–43 (1983).
  72. P. Winston, Learning Structural Descriptions from Examples, Ph.D. Dissertation, Report MAC-TR-76, MIT, Cambridge, MA.
  73. S. C. Salveter, "Inferring conceptual graphs," *Cog. Sci.* **3**, 141–166 (1979).
  74. R. C. Schank, *Dynamic Memory*, Cambridge University Press, New York, 1982.
  75. N. Haas and G. G. Hendrix, Learning by Being Told, in R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.), *Machine Learning*, Tioga, Palo Alto, CA, pp. 405–427, 1983.
  76. N. D. Roussopoulos, A Semantic Network Model of Data Bases, Ph.D. Thesis, University of Toronto, Department of Computer Science, 1976.
  77. G. G. Hendrix, "Expanding to utility of semantic networks through partitioning," *Proc. of the Fourth IJCAI*, Tbilisi, Georgia, 1975, pp. 115–121.
  78. G. E. Heidorn, Natural Language Inputs to a Simulation Programming System, Report NPS-55HD72101A, Naval Postgraduate School, Monterey, 1972.
  79. G. E. Heidorn, Augmented Phrase Structure Grammar, in R. C. Schank and B. L. NashWebber (eds.), *Theoretical Issues in Natural Language Processing*, ACL, pp. 1–5, 1975.
  80. K. Jensen and G. E. Heidorn, The Fitted Parse: 100% Parsing Capability in a Syntactic Grammar of English, *Proceedings of the Conference on Applied Natural Language Processing*, Santa Monica, CA, pp. 93–98, 1983.
  81. J. F. Sowa and E. C. Way, "Implementing a semantic interpreter using conceptual graphs," *IBM J. Res. Devel.* **30**(1) (1986).
  82. S. E. Fahlman, *NETL: A System for Representing and Using Real-World Knowledge*, MIT Press, Cambridge, MA, 1979.
  83. D. L. Waltz and J. B. Pollack, "Massively parallel parsing," *Cog. Sci.* **9**, 51–74 (1985).

### General References

- National Physical Laboratory, *International Conference on Machine Translation of Languages and Applied Language Analysis*, Her Majesty's Stationery Office, London, 1961.
- M. C. McCord, "Slot grammars," *Am. J. Computat. Ling.* **6**(1), 31–43 (1980).
- R. C. Schank and C. K. Riesbeck, (eds.), *Inside Computer Understanding*, Erlbaum, Hillsdale, NJ, 1981.
- M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, 1968.

J. SOWA  
IBM

### SEMANTICS

The term "semantics" means meaning or the study of meaning. The meaning of a natural-language word or sentence is the entity or action it denotes. In computer understanding of language, semantic interpretation is the process of determining the meaning of the input. This presupposes that one has, first, a computational representation for meaning, and, second, a method for deriving the representation for a given input. The adequacy of any particular representation or interpretation method will depend on the complexity of the particular application; these issues are discussed below.

### Relationship between Semantics and Syntax

Semantic interpretation of all but the shallowest level (as, e.g., in the ELIZA (qv) system) takes place in association with a syntactic analysis that determines the structure of the sentence (see Parsing). It is a matter of controversy as to whether

the two kinds of analysis, syntactic and semantic, are better performed by separate processes or better mixed together in one process. Separation is more usual, except in systems dealing only with syntactically simple input (see also Semantic grammars). It is clear, however, that syntactic knowledge interacts with semantic knowledge (just as it does with other forms of linguistic and world knowledge); semantic analysis requires information about the structure of the input sentence, but in order to provide this, the parser requires information about the meaning of constituents. This is because many sentences are structurally ambiguous, i.e., they have more than one possible structure, and the choice is determined by meaning. For example, in

Nadia washed the dog with long hair [1]

the parser could take "with long hair" as part of the description of the dog or as the method by which the dog was washed (as in "Nadia washed the dog with pet shampoo"); a semantic process must tell the parser which structure leads to a sensible interpretation. Some early systems (e.g., Ref. 1) were based on the premise that syntax is an artifact that plays little or no role in semantic analysis; it is now generally accepted that this is incorrect. Marcus (2) has shown that in the most general case a complete semantic interpretation necessitates a complete syntactic analysis, although this need not always be true in simple applications; and since syntactic and semantic knowledge are relatively independent, proper programming methodology suggests keeping the two in separate modules.

If it is decided that the two processes, syntactic and semantic, are to be kept separate, then ideally they should proceed in parallel and communicate with one another as necessary. An alternative (used in the LUNAR system; see below) is that semantic interpretation may follow parsing but be allowed to reject the output of the parser as anomalous and force it to suggest something different; the disadvantage of this, however, is that the semantic interpreter is forced to make a decision on semantic well-formedness without knowing what alternatives there are, possibly accepting only the second best.

### Semantic Theories

The nature of meaning and means for its representation are also topics of interest in linguistics and the philosophy of language. The concerns of scholars in these fields are, however, different from those of AI; rather, the emphasis is on describing the basis for language and meaning. A "semantic theory" that is adequate for AI may not be adequate as an explication of the nature of reference, and a theory adequate from a philosophical viewpoint may not be usable in AI. Nevertheless, there is common ground, and the AI practitioner should be aware of the other perspective.

One particular concern in all semantic theories is compositionality. The principle of compositionality is that the meaning of the whole sentence is some systematic function of the meaning of its components. This is intuitively reasonable, but there are two alternatives. The first is that the meaning of the whole is a systematic function of not only the meaning of the parts but also of the context and situation in which it is uttered. Ideally, a semantic theory should be able to account for this as well. The second alternative is that the meaning of the whole is a non-systematic function of the meaning or form of the parts; this seems counterintuitive but in fact is a property of procedural semantics, discussed below.

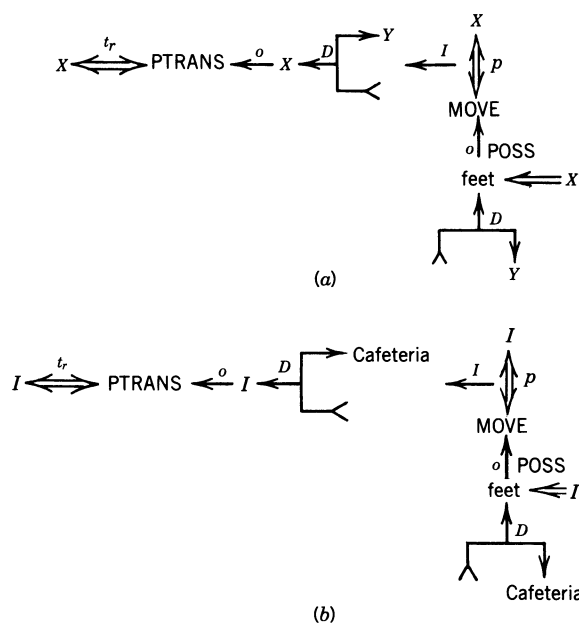
### Decompositional Semantics

Theories of decompositional semantics attempt to represent the meaning of each word by decomposing it into a set of semantic primitives. For example, the word "chair" could be represented as follows:

(Object), (Physical), (Nonliving), (Artifact),  
(Furniture), (Portable), (Something with legs), [2]  
(Something with a back), (Something with a seat),  
(Seat for one)

In practice, such a lexical decomposition program is problematic. It is extremely difficult, perhaps even impossible in principle, to find a suitable, linguistically universal collection of semantically primitive elements in which all words (of all languages) could be decomposed into their necessary properties. For example, not all the properties shown in [2] are defining features of a chair, or are even present in all chairs, and yet they seem necessary in distinguishing chairs from other objects. Even simple words whose meanings seem straightforward often turn out to be extremely difficult to characterize. Decompositional semantics is also problematic in its notion of how a sentence is represented; it is necessary to somehow combine such lexical representations into the representation of the whole sentence. And there must then be methods for inference (qv) in context upon the resultant structure.

There have been attempts in AI to make decompositional semantics usable by adding these missing elements to the theory. Notable among these are the preference semantics system of Wilks (3), the early conceptual-dependency (qv) representations of Schank (4) and Sowa's conceptual graphs (5). In particular, Schank had a systematic method for building his primitives into a structure that represented a sentence and for performing inference upon it. For example, the lexical entry for the verb "to walk" might be the conceptual-dependency diagram shown in Figure 1a. This may be glossed as "X physi-



**Figure 1.** (a) Lexical entry for "walk" in conceptual dependency. (b) Representation of "I walked to the cafeteria" in conceptual dependency. (Adapted from R. Schank, *Identification of Conceptualizations Underlying Natural Language*, in Ref. 4. Courtesy of W. H. Freeman and Company, San Francisco.)

cally moved itself to place *Y* by moving *X*'s feet to *Y*"; the act of walking has been decomposed into semantic primitives. This structure can be built into the representation of the meaning of a sentence; thus "I walked to the cafeteria" could be represented as in Figure 1*b* (see also Parsing, expectation-driven). However, the set of conceptual-dependency primitives is incomplete and unable to capture nuances of meaning. More recently, Sowa (5) has proposed a system of conceptual graphs that, by combining a compositional approach with an approach similar to that of frames (qv), attempts to gain the advantages of both.

### Montague Semantics

In his well-known "PTQ" paper (6), Montague presented the complete syntax and semantics for a small fragment of English. Although it was limited in vocabulary and syntactic complexity, Montague's fragment dealt with many previously unsolved semantic problems. Montague's formalism is exceedingly complex, and it is not presented here; instead, a discussion of the formalism's important theoretical properties is presented. The reader interested in the details will find Ref. 7 a useful introduction.

Montague's theory is truth-conditional and model-theoretic. Truth-conditional means that the meaning of a sentence is the set of necessary and sufficient conditions for the sentence to be true, i.e., to correspond to a state of affairs in the world. Model-theoretic means that the theory uses a formal mathematical model of the world in order to set up relationships between linguistic elements and their meanings. Thus, semantic objects are entities in this model, namely individuals and set-theoretic constructs defined on entities. Since sentences are not limited to statements about the present world as it actually is, Montague employs not just one model of the world but rather a set of possible worlds. The truth of a sentence is then relative to a chosen possible world and point in time; a sentence could be true in one world at a particular time and false at another time or in a different possible world. A possible world-time pair is called an index.

Montague takes the word to be the basic unit of meaning, assuming that for each index there is an entity in the model for each word of the language. The same entity could be represented by more than one word, of course: Thus, in some possible world the words "unicorn" and "pigeon" could denote the same set of individuals—in particular, they could both denote the empty set.

For Montague, then, semantic objects, the results of the semantic interpretation, are such things as individuals in (the model of) the world, individual concepts (which are functions to individuals from the set of indexes), properties of individual concepts, and higher-order functions. At the top level the meaning of a sentence is a truth condition relative to an index. These semantic objects are represented by expressions of an intensional logic; i.e., instead of translating English directly into these objects, a sentence is first translated to an expression of intensional logic for which, in turn, there exists an interpretation in the model in terms of these semantic objects. However, the intensional logic is merely a convenience in specifying the translation and can be eliminated from the theory without ill effect (except in perspicuity).

Montague's system contains a set of syntactic rules and a set of semantic rules, and the two are in one-to-one correspondence. Each time a particular syntactic rule applies, so does the corresponding semantic rule; whereas the one operates on

some syntactic elements to create a new element, the other operates on the corresponding semantic objects to create a new object that will correspond to the new syntactic element. Thus, the two sets of rules operate in tandem.

The typical semantic rule involves functional application: The translation of the new constituent is formed by applying the translation of one of its components, as a function, to the other, as an argument to the function. Because of the tandem operation of the rules, a strong typing applied to semantic objects, and the fact that the output of a semantic rule is always a systematic function of its input, Montague semantics is compositional.

Although Montague semantics has much to recommend it, it is not possible to implement it directly in a practical natural-language understanding (NLU) system for two reasons. The first is that a direct implementation of it would be computationally impractical. Montague semantics throws around huge sets, infinite objects, functions of functions, and piles of possible worlds with great abandon. The second is that truth-conditional semantics is inadequate for AI. AI is interested not so much in whether a state of affairs is or could be true in some possible world but rather in the state of affairs itself; thus, AI needs systems in which semantic objects are objects or expressions in a declarative or procedural knowledge-representation system (see Representation, knowledge; Representation, procedural), i.e., systems in which the content of the sentence itself is retained and is represented in a convenient symbolic manner.

There have, however, been attempts to take the intensional logic that Montague uses an intermediate step in his translations and give it a new interpretation in terms of AI-type semantic objects, thus preserving all other aspects of Montague's approach (8); there has also been interest in using the intensional logic itself (or something similar) as an AI representation (9) even though it is not relevant to the substance of Montague's theories. But although it may be possible to make limited use of intensional-logic expressions, there are many problems that need to be solved before intensional logic or other flavors of higher-order logical forms could support the type of inference (qv) and problem solving (qv) that AI requires of its semantic representations; see Refs. 9 and 10 for a useful discussion.

Nevertheless, it is possible to use many aspects of Montague's approach to semantics in AI; for example, the Absity system (see below) is described by its author as "Montague-inspired."

### Situation Semantics

A recent semantic theory, and one still under development, is situation semantics, first presented by Barwise and Perry (11). The viewpoint taken is that, as mentioned above, truth-conditional semantics is inadequate because if the ultimate denotation of a sentence is the value true or false in a possible world, the content of the sentence itself has been lost. Moreover, in truth-conditional semantics all logically equivalent statements have the same truth conditions even if their content differs; thus, the following two sentences, both tautologies, would not be semantically distinguished even though they say quite different things:

East is east and west is west [3]

Either Nadia is going to be on time or she isn't [4]

Situation semantics attempts to formalize the idea of a situation in the real world as a suitable semantic object; indeed, Barwise and Perry see linguistic semantics as just a special case of meaning in the world. The two examples above would refer to two quite different situations. Situation semantics has been held to have much promise for application in computational language understanding. However, at this writing it has yet to be applied in any significant AI work.

### Procedural Semantics

An early approach to semantic interpretation, one that has been extremely influential, is Woods's procedural semantics (12,13) (qv). In this approach, production rules (see Expert systems; Rule-based systems) translate the parsed input into procedure calls that operate upon a database, and the meaning of a sentence is identified with the corresponding procedure call. For example, the following translations may be made in a natural-language front-end for an airline reservation system (12,13):

- AA-57 is nonstop from Boston to Chicago.  
equal (numstops (aa-57, boston, chicago), 0) [5]
- Every flight that leaves Boston is a jet.  
(for every X1/flight: depart (X1, boston); jet (X1)) [6]
- What is the departure time of AA-57 from Boston?  
list (dtime (aa-57, boston)) [7]
- Does AA-57 belong to American Airlines?  
test (equal (owner (aa-57), american-airlines)) [8]

The representation is essentially equivalent to first-order logic (qv). Note that the system accepted only questions for retrieval from the database, not declarative sentences for updating it, a point that is discussed below. Nevertheless, for simplicity, Wood's style of showing most example sentences in a "raw" declarative form is used rather than an interrogative form. The representation of the interrogative form was to take the declarative form as an argument to the procedures **test** (for yes-no questions) or **list** (for "wh-" questions), which formatted the output accordingly. For example, in [8] the procedure **equal** checks for the equality of its arguments and returns **true** or **false**; the procedure **test** will then present the result to the user as "yes" or "no."

Procedural semantics was used in LUNAR (qv) (14), also known as LSNLIS, a system for accessing in English a database of information about rock samples collected on the Apollo lunar missions. LUNAR was one of the earliest database interfaces with a significant natural-language ability, and it greatly influenced work in the area. Procedural semantics has been used in one form or another in a large number of systems.

**Problems of Procedural Semantics.** The problems with procedural semantics are that it is ad hoc and noncompositional. For example, in Woods's original system there was no consistent interpretation of the verb "depart":

- AA-57 departs from Boston.  
depart (aa-57, boston) [9]
- AA-57 departs from Boston to Chicago.  
connect (aa-57, boston, chicago) [10]
- AA-57 departs from Boston at 8:00 a.m.  
equal (dtime (aa-57, boston), 8:00am) [11]

This variation is possible because the rules are both very specific and very powerful. Rules often look for specific words as their trigger, and if the word in the input were changed to a synonym, the rule would not fire. For example, the rules for the sentences above look for the verb "depart" even though the verb itself is not used in the output. Similarly, each rule applies only to a particular sentence structure; if the same meaning can be expressed with more than one sentence structure, there must be a separate rule for each. Moreover, the output is not tied in any way to the input; a rule can ignore any or all of its input or make changes that are quite inconsistent with those of other rules; nor is there anything preventing the construction of rules that would result in conjunctions with conflicting terms. A particular set of rules could probably be made reasonably compositional by appropriate adjustment of the procedure calls into which sentences are translated. However, the system would still not be compositional by design, and it would be easy to inadvertently lose compositionality again when extending the system.

Adding an ability to update the database would also be antithetical to compositionality in the system, for then either the meaning of a procedure would have to vary with context or the translation of the whole sentence would have to vary with sentence form. To see the problem, consider the sentence

AA-57 is nonstop from Boston to Chicago. [12]

Previously, the "raw" meaning of this sentence was said to be

equal (numstops (aa-57, boston, chicago), 0) [13]

and that therefore the meaning of the interrogative form,

Is AA-57 nonstop from Boston to Chicago? [14]

is

test (equal (numstops (aa-57, boston, chicago), 0)) [15]

But if sentence [12] is to be allowed as input to modify the database, its translation must be more carefully considered. Possibilities include its "raw" form, [13], and a form analogous to [15], such as

assert (equal (numstops (aa-57, boston, chicago), 0)) [16]

But in either case the meaning of **equal** has suddenly changed; instead of being a predicate, it is now an assignment statement. The alternative is to translate [15] into an entirely different procedure call:

make-equal (numstops (aa-57, boston, chicago), 0)) [17]

The choice is thus from three unpalatable situations: One is to say that **equal** shall somehow be sensitive to the context in which it is called and adjust its behavior accordingly (a dubious idea both semantically and from a programming viewpoint); the second is to have the context, **test** or **assert**, modify its argument as necessary; the third is to double the number of rules and predicates needed in the system, having one set for interrogative forms and another for declaratives, again defeating compositionality. It may be possible to circumvent this problem by the use of a PROLOG-like database language (see Logic programming); such languages have the same procedural-declarative ambiguity and resolve the ambiguity in context:

:- equal (x, y) [18]

equal (x, y) :- [19]

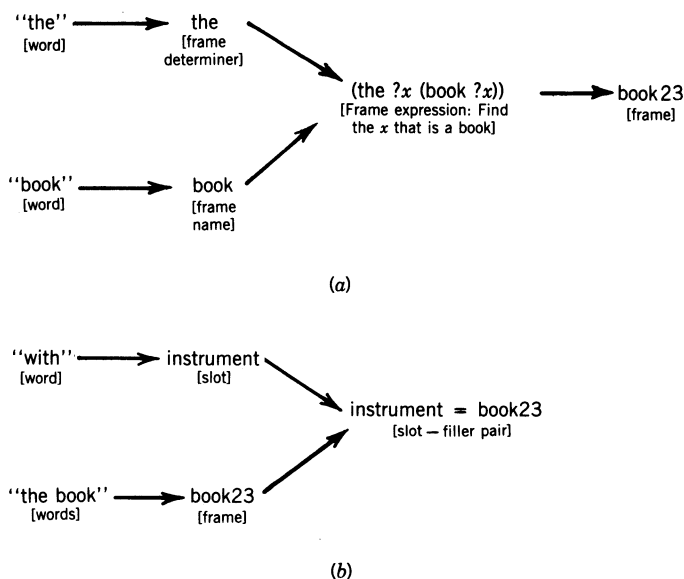
Another problem with this approach is that semantic interpretation generally occurs after the parsing of the sentence is complete, and so the interpretation of the first part of the sentence is not yet available to aid the parser if a structural ambiguity arises later in the sentence; rather the parser must be forced to back up if semantic anomalies occur. In addition, because the interpretation of the sentence itself proceeds bottom-up (see Processing, bottom-up and top-down) but not left to right, the resolution of intrasentential reference is problematic; LUNAR, e.g., could resolve many pronoun references to previous sentences but not references to things mentioned in the same sentence.

**Procedural Semantics as a Semantic Theory.** The status of procedural semantics itself as a theory of semantics has been a matter of considerable controversy. There are many variations, but the gist of procedural semantics as a semantic theory is that the meaning of a sentence is the procedure into which the sentence is compiled, either in the computer or in the mind; the procedure itself can be seen as the intension of the sentence, the concept or idea behind it, and the result of the execution as the extension, the particular entity denoted. [Woods himself would not agree with this; he argues that truth-conditionality must also play a role (15).]

The notion of a procedure is so basic to computation that procedural semantics seems a very natural approach for AI, and it has been used in many systems since LUNAR. Since its original incarnation it has been refined considerably and is still today perhaps the predominant approach, despite its problems and its essentially ad hoc nature. However, in its pure form as described above, procedural semantics is not adequate for AI because the procedures themselves do not have an adequate interpretation and the items they manipulate are uninterpreted symbols. This is not a difficulty if one is just inserting or retrieving database values with little or no interpretation (this is, of course, an important application), but if one is interested in maintaining and manipulating a knowledge base, performing inference, solving problems, and the like, procedural semantics suffers from the problem that symbols have been translated into other symbols, but the new symbols are scarcely easier to deal with semantically than the old ones.

### Knowledge-Base Semantics

**Outline.** Knowledge-base semantics is an approach that avoid many of the problems of decompositional and procedural semantics. An AI-style knowledge-representation formalism (see Representation, knowledge) is taken as the meaning representation, with the elements of the representation regarded as semantic objects that provide an interpretation of the constituents of the input sentence. For example, in Hirst's Absity system (16) nouns are identified with the frames of a frame-based representation (qv); the word "book," e.g., is listed in the system's dictionary as a reference to the particular frame that describes books. Similarly, verbs are also frames, determiners such as "the" and "a" are frame selectors (knowledge-base retrieval functions), and prepositions are slots. After individual words are interpreted as objects in the representation, they are combined to create larger semantic objects that correspond to the syntactic constituents in which they are contained. Thus, the noun phrase "the book" is formed from the frame for "book" and a frame selector for "the," the result being an instance of a frame, namely the instance, say book23,



**Figure 2.** (a) Interpreting "the book" in Absity. (b) Interpreting "with the book" in Absity.

of the book frame that is presently in focus (see Discourse understanding); see Figure 2a. Similarly, prepositional phrases are filled slots; if "with" is interpreted as the instrument slot of a particular action, then "with the book" will be the slot-and-filler pair instrument = book23 (Fig. 2b). At the top level a declarative sentence is interpreted as a statement in the representation to add some information—the meaning of the sentence—to a knowledge base, and a question is interpreted as a statement that retrieves information from the knowledge base.

Knowledge-base semantics can thus be compositional: The meaning of the whole can always be a systematic function of the meaning of the parts, and it relies on the representation formalism being systematic enough to permit this. (This is not, however, a necessary property of such systems.) The approach has echoes of Montague semantics: The knowledge base plays the role of the model, and each syntactic constituent corresponds to an object in the model, the objects being constructed compositionally. The notion of truth relative to a possible world, however, is replaced by a call on the knowledge base; that is, unlike the Montague approach, content is retained at the top level.

The advantages of knowledge-base semantics in the resolution of ambiguity (see below) are obtained only if semantic interpretation proceeds in parallel with syntactic analysis, so that later syntactic analysis may make use of the interpretation of the earlier part of the sentence. This is seen in its strongest form in Absity, in which syntax and semantics move in lockstep; each time a syntactic rule applies upon some syntactic constituents, the corresponding semantic rule applies to the corresponding semantic objects. The semantic rule is usually no more than constructing a new semantic object in the obvious way from the ones given. In contrast, the PSI-KLONE system of Bobrow and Webber (17) "cascades" the syntactic and semantic analysis.

**Advantages and Disadvantages of Knowledge-Base Semantics.** Because partial results are always well-formed objects in the knowledge representation, they can be used in retrieval and inference even as semantic interpretation proceeds. In



particular, this means that information from the knowledge base can be used to resolve ambiguities. For example, in the case of sentence [1] above, the knowledge base can be asked which attachment of "with long hair" makes more sense, and it may use whatever methods it has available to it to answer the question. Hirst (16) describes disambiguation methods for use in knowledge-base semantics both for structural ambiguities like [1] and for lexical ambiguities, in which there is more than one frame to which a particular word (such as "bank"—a financial institution or the edge of a river) may be mapped. The latter is especially important for disambiguating prepositions, as most prepositions are highly ambiguous as to the slot they denote. A disadvantage of this approach is that it assumes a compositionality in the representation that may not be achievable. For example, no present system can represent

They even stole the bathtub [20]

as a composition of the representations of "They stole the bathtub" and "even"; indeed, it is not clear how a word like "even" should be handled at all.

**Comparison with Procedural Semantics.** Knowledge-base semantics and procedural semantics are similar in that both interpret sentences as calls to insert or retrieve information. The difference is that procedural semantics never interprets the symbols it uses; it merely constructs a string that, when complete, is a database call. Partial results are not, in general, well-formed semantic objects in their own right and cannot be used in knowledge-base retrieval or inference. Such systems may have no dictionary as such; all the knowledge—both of words and of the world—used in the interpretation is encoded in the production rules. Knowledge-base semantics, on the other hand, is able to use the full services of a retrieval and inference system when necessary. It is thus more powerful but also less self-contained; it is more suited to use in a large, general AI system and less to use in a natural-language interface to a database or other non-AI application.

## BIBLIOGRAPHY

1. C. K. Riesbeck, Conceptual Analysis, in R. C. Schank (ed.), *Conceptual Information Processing Fundamental studies in computer science* 3, North-Holland, Amsterdam, pp. 83–156, 1975.
2. M. P. Marcus, Some Inadequate Theories of Human Language Processing, in T. G. Bever, J. M. Carroll, and L. A. Miller (eds.), *Talking Minds: The Study of Language in Cognitive Science*, MIT Press, Cambridge, MA, pp. 253–278, 1984.
3. Y. A. Wilks, "A preferential, pattern-seeking semantics for natural language inference," *Artif. Intell.* **6**, 53–74 (1975).
4. R. Schank, "Identification of conceptualizations underlying natural language," in R. Schank and K. M. Colby (eds.), *Computer Models of Thought and Language*, Freeman, San Francisco, CA, pp. 187–247, 1973.
5. J. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*, The systems programming series, Addison-Wesley, Reading, MA, 1984.
6. R. Montague, The Proper Treatment of Quantification in Ordinary English, in K. J. J. Hintikka, J. M. E. Moravcsik, and P. C. Suppes (eds.), *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, Reidel, Dordrecht, pp. 221–242, 1973; also in R. H. Thomason (ed.), *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press, New Haven, CT, pp. 247–270, 1974.

7. D. R. Dowty, R. E. Wall, and S. Peters, *Introduction to Montague Semantics*, Synthese language library 11, Reidel, Dordrecht, 1981.
8. J. R. Hobbs and S. J. Rosenschein, "Making computational sense of Montague's intensional logic," *Artif. Intell.* **9**(3), 287–306 (December 1977).
9. R. C. Moore, Problems in Logical Form, *Proceedings of the Nineteenth Annual Meeting of the Association for Computational Linguistics*, Stanford, July 1981, pp. 117–124; also Technical Note 241, Artificial Intelligence Center, SRI International, April 1981.
10. D. S. Warren, Using  $\lambda$ -Calculus To Represent Meanings in Logic Grammars, *Proceedings of the Twenty-First Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA, June 1983, pp. 51–56; also Technical Report 83/045, Department of Computer Science, State University of New York Stony Brook, January 1983.
11. J. Barwise and J. R. Perry, *Situations and Attitudes*, MIT Press, Bradford Books, Cambridge, MA, 1983.
12. W. Woods, "Procedural semantics for a question-answering machine," *AFIPS Conf. Proc.* **33** (Fall Joint Computer Conference), pp. 457–471 (1968).
13. W. Woods, *Semantics for a Question-Answering System*, Outstanding dissertations in the Computer Sciences, Garland, New York, 1979.
14. W. Woods, R. M. Kaplan, and B. L. Nash-Webber, The Lunar Sciences Natural Language Information System: Final Report, Report 2378, Bolt, Beranek and Newman, Cambridge, MA, June 15, 1972.
15. W. Woods, Procedural Semantics as a Theory of Meaning, in A. K. Joshi, B. L. Webber, and I. A. Sag (eds.), *Elements of Discourse Understanding*, Cambridge University Press, Cambridge, UK, pp. 300–334, 1981.
16. G. Hirst, *Semantic Interpretation and the Resolution of Ambiguity*, Studies in natural language processing, Cambridge University Press, 1987.
17. R. J. Bobrow and B. L. Webber, PSI-KLONE: Parsing and Semantic Interpretation in the BBN Natural Language Understanding System, *Proceedings of the Third Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, Société canadienne pour l'étude de l'intelligence par ordinateur, Victoria, B.C., pp. 131–142, May 1980.

G. HIRST  
University of Toronto

## SEMANTICS, PROCEDURAL

The term "procedural semantics" (see Semantics) was coined in 1968 by this author (1) to describe a methodology for characterizing the meanings of sentences for computer systems that answer natural-language questions (see Question answering). In this methodology procedures were used to characterize the truth conditions of statements as well as those used for answering questions and carrying out actions. Later, Winograd (2,3) used similar terminology to refer to a methodology for expressing procedures for natural-language parsing and interpretation as well as subsequent response. Here "procedural" specifications were viewed as an alternative to formal specifications such as grammars and semantic interpretation rules (see Grammar, augmented-transition-network).

These two uses of the term are quite different and have each generated their own controversies. Shortly after Winograd's thesis a brief controversy arose in which the relative merits of expressing AI programs in terms of notations that

are overtly procedural were debated as opposed to using "declarative" representations, which look more like specifications in formal logic (qv) are less encumbered with specific details as to how the computation will be performed, and can in principle be used in different ways by different "interpreters." What was really going on in these debates, although poorly expressed, was an attempt to articulate the advantages of having an abstraction in which the essence of a phenomenon can be expressed without an unnecessary commitment to nonessential details of an implementation. The term "declarative" is still frequently used to describe representations that can be viewed in terms of what they say or assert rather than the details of the computations they will cause.

The original use of the term "procedural semantics," however, and the one to which this entry is addressed, refers not to the appearance of the representation but to its underlying semantic foundation. The representations in Ref. 1 were formal extensions of the predicate calculus (see Logic, predicate) that could easily qualify as declarative representations. However, the term "procedural" was used in describing their semantics to emphasize that even the notations of formal logic derive their meaning from a kind of abstract computation. That is, the logical operations of AND, OR, and IF-THEN derive their meanings from rules for assigning truth values to complex sentences as a function of the truth values of their constituent clauses. Similarly, the operations FOR-EACH and THERE-IS derive their meanings from rules for assigning truth values to universally and existentially quantified sentences. Thus, one can interpret expressions in a formal logic as a kind of abstract procedure for determining truth values. From this perspective even declarative representations have a procedural foundation that determines their meaning.

The point of the procedural semantics perspective is that logical notations, when viewed as abstract procedures, can be generalized to account for a more diverse range of phenomena than has been formalized by the first-order predicate calculus. In particular, procedures can formalize the meanings of actions that have causal connections to the real world (such as the control of a robot vehicle) (see Autonomous vehicles; Reasoning, causal; Robots, mobile). Viewed as abstract procedures, the mechanisms of formal logic can be extended to deal with interrogative and imperative sentences in addition to declarative propositions. Moreover, generalizations of the notion of universal and existential quantification can be defined as abstract procedures to allow appropriate models for such things as the definite determiner "the" and the quantifier "most." Within this framework, a powerful range of quantificational structures has been developed, including definite determiners, operators for finding antecedents of pronouns, and general counting, filtering, and averaging operators (4).

From this overall perspective work in procedural semantics splits into two distinct areas of focus: techniques for efficient implementation of computerized natural-language understanding (qv) systems and search for a philosophical theory of meaning capable of accounting for the range of phenomena that people traditionally describe by the word. The former has produced powerful techniques for building natural-language interfaces (qv) (1,4-6) and has stimulated related research such as the query-optimization work of Reiter (7) and the theory of anaphoric reference of Webber (8). The theoretical perspective is embodied in a series of articles (9,10-12), which has addressed the more subtle philosophical problems of developing an adequate theory of meaning.

As a semantic theory, procedural semantics attempts to provide a philosophically adequate foundation for meaning that deals with the causal connections that relate internal reasoning representations and logical formalisms to objective situations in the world. It postulates that the meanings of expressions reside in abstract procedures that can sometimes be invoked to manipulate a combination of internal mental representations and (through the sensory motor system) the external physical world. This requires procedures that are abstract in the sense that they do not express nonessential implementational commitment (10,12) but that, unlike abstract mappings from possible worlds into truth values, can be finitely represented and used by an organism or a computer system. Procedural semantics builds on the insight from computability theory and formal automata theory that infinite classes of objects can be finitely characterized by means of an abstract automaton that recognizes or generates instances of the class.

In the most refined version of the theory (10,12) a distinction is made between the external language one speaks and the internal language/notation/representation with which one reasons and within which one interprets the meanings of external utterances. This internal language itself needs a semantics, and it is this problem that procedural semantics addresses. The theory distinguishes between "meaning functions," the procedures that characterized the meaning of a term, and "recognition functions," the procedures routinely used to recognize instances. The former determine ground truth but may be difficult, or in some cases impossible, to execute (e.g., through lack of access to the appropriate data). These meaning functions nevertheless are available as abstract objects, capable of manipulation by reasoning processes, to characterize the meaning of a term. They are used in cases of importance to calibrate the recognition functions and to discover violations to conjectured hypotheses. The point of this distinction is that although any reasoning entity will necessarily have procedures it will use to interpret sensory input and draw conclusions, these procedures are not the ones that fill the role of "meaning." Without a distinction between a meaning criterion and the procedures that draw conclusions, it is not possible to interpret conclusions as making claims or predictions beyond the data actually examined to deduce them. Without such a distinction, it is not possible to learn improved models of reality by detecting violations of predictions and altering the models appropriately.

## BIBLIOGRAPHY

1. W. A. Woods, Procedural Semantics for a Question-Answering Machine, *AFIPS Conference Proceedings, Fall Joint Computer Conference*, Montuole, NJ, pp. 457, 471, 1968.
2. T. Winograd, Procedures as a Representation for Data in a Computer Program for Understanding Natural Language, MAC TR-84, Ph.D. Thesis, MIT, Cambridge, MA, 1971.
3. T. Winograd, A Procedural Model of Language Understanding, in R. Schank and K. Colby (eds.), *Computer Models of Thought and Language*, W. H. Freeman, San Francisco, pp. 152-186, 1973.
4. W. A. Woods, Semantics and Quantification in Natural Language Question Answering, in M. Yovitz (ed.), *Advances in Computers*, Vol. 17, Academic Press, New York, pp. 2-64, 1978; also in B. J. Grosz, K. S. Jones, and B. L. Webber (eds.), "Readings in Natural Language Processing," Morgan-Kaufmann, Los Altos, CA, pp. 205-248, 1986.

5. W. A. Woods, Progress in Natural Language Understanding: an Application to Lunar Geology, *AFIPS Conference Proceedings*, Vol. 42, 1973 National Computer Conference, AFIPS, Montvale, NJ, pp. 441-450, 1973.
6. W. A. Woods, R. M. Kaplan, and B. L. Nash-Webber, The Lunar Sciences Natural Language Information System: Final Report, BBN Report No. 2378, Bolt Beranek and Newman, Inc., Cambridge, MA, June, 1972. (Available from NTIS as N72-28984.)
7. R. Reiter, An Approach to Deductive Question-Answering, Report No. 36499, Bolt Beranek and Newman, Cambridge, MA, October 1977.
8. B. L. Webber, *A Formal Approach to Discourse Anaphora*, Garland, New York, 1978.
9. W. A. Woods, *Semantics for a Question-Answering System*, Garland, New York, 1979.
10. W. A. Woods, Procedural Semantics as a Theory of Meaning, in A. K. Joshi, B. L. Webber, and I. A. Sag (eds.), *Elements of Discourse Understanding*, Cambridge University Press, London, pp. 300-334, 1981.
11. W. A. Woods, Under What Conditions can a Machine use Symbols with Meaning, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 47-48, 1983.
12. W. A. Woods, Problems in Procedural Semantics, in Z. Pylyshyn and W. Demopoulos (eds.), *Meaning and Cognitive Structure*, Ablex, Norwood, NJ, pp. 55-85, 1986.

W. A. WOODS  
Applied Expert Systems, Inc.  
and Harvard University

## SENSORS

Robots are machines capable of interacting with the physical world in a seemingly intelligent manner. Intelligent interaction implies the ability to transduce and understand the environment and to formulate behavior appropriate for the situation at hand. A key element in this process is the sensing of the characteristics of the environment.

This entry provides an overview of the various sensing modalities and the associated sensors available to robots. A classification system is developed and presented as a possible framework in which to discuss robotic sensory systems.

### Taxonomy of Robot Sensory Information

**Direct/Indirect.** One possible classification characteristic for sensors is the modality employed by the device. Sensation may be divided into several general classes corresponding to the physical quantities or qualities transduced or information provided by the sensor. In the biological domain these include sight, hearing, taste, touch, and smell. In general, there are several types of sensors that transduce the same physical quantity. For example, the distance between the robot and an object in its environment could be determined through the use of stereo vision in the visible spectrum, laser rangefinding, or acoustical techniques. The sensors corresponding to those three techniques differ greatly in their design and the underlying physics of their operation; yet they can provide the same information to the robot. This leads to a classification by transduction method. The extent of physical interaction between the sensor and the object whose qualities are being transduced may also be used as a classification parameter.

Sensory information may be broken into two general classes: that which is obtained **DIRECTLY** by the robot and that which is obtained **INDIRECTLY**, (see Fig. 1). **DIRECT** sensory information is provided by sensors/systems local to the robot and may include, e.g., visual and tactile information. **INDIRECT** information is provided to the machine by external systems and could include information regarding the relative positions of other machines that are out of the range of the robot's own sensing systems. The **DIRECT** vs. **INDIRECT** classification deals more with the source of the information than with its content since most, if not all, forms of information can either be gathered by the robot's systems or supplied to the robot from an outside source. The purpose of this distinction is to differentiate between information obtained by sensors local to the robot system and that obtained remotely. Thus, when considering sensing systems for robots, only those sensors providing **DIRECT** information are considered. In practice, the factors determining whether a specific sensor is located on (in) each robot system (**DIRECT**) or whether they are located external to the robot (**INDIRECT**) are usually the cost of the sensor, the amount of processing required, the proximity of the robot to external sensing systems, and the bandwidth of the data path between the robot and the external systems; e.g., information regarding the velocity of a Mars rover must be available directly to the rover as opposed to having this information transmitted from an Earth station. Described in this entry are the sensors that are directly usable by the robot system.

**Internal/External.** Sensory information may be further categorized according to whether it concerns events or quantities that are **INTERNAL** or **EXTERNAL** to the robot. As in biological systems, **INTERNAL** sensing provides information regarding the state of the organism itself: quantities such as the forces present at various joints, the velocities of the limbs, temperature of the organism, and the status of various subsystems, e.g., the circulation system. Analogous quantities are available to robots, such as the temperatures of the various motors and circuits, and subsystems, including the bus(es), processors, drive systems, etc. **EXTERNAL** information concerns the environment in which the robot exists and its relation to the robot, e.g., the locations and orientations of objects and the motion of the robot relative to the environment.

**Global/Local.** The final two levels in the hierarchy presented in Figure 1 discuss the spatial and temporal extents of the characteristic transduced. The first level provides a distinction between **GLOBAL** and **LOCAL** information.

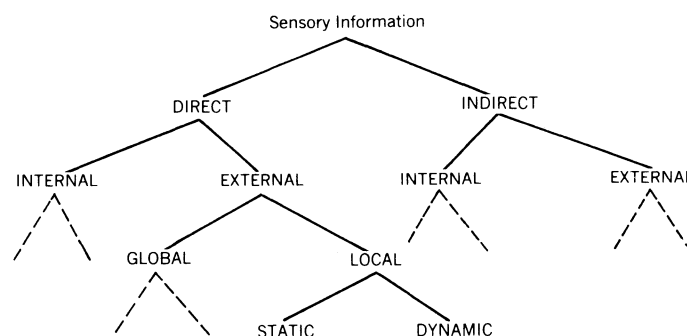


Figure 1. Taxonomy of sensory information.

GLOBAL sensory information refers to information that concerns specific nonspatial characteristics of the environment. The temperature of the atmosphere and the overall brightness of the environment are two examples of GLOBAL information. LOCAL information rules the properties or characteristics of a specific element or location in the environment. The temperature of an object and the color of a particular region in the visual field are examples of LOCAL information.

**Static/Dynamic.** The final level in the taxonomy is based on the temporal characteristics of the stimulus being sensed. The classification parameters of STATIC and DYNAMIC are used.

STATIC sensory information includes information about invariant characteristics of the object or environment. This includes information both about characteristics that are static, e.g., the mass of an object, and unchanging relationships between the stimulus and the sensing system, e.g., the point of contact of unmoving objects.

The other companion category at this level of the taxonomy includes DYNAMIC information. The emphasis here is on information regarding the changes between the sensory system and the stimulus. This information includes both changes in the object, e.g., swelling or deformation, and changes in the relationship between the object and the sensing system, e.g., movement.

In summary, this classification scheme begins by differentiating between information supplied by the robot's sensory system and that supplied by external sensors (DIRECT vs. INDIRECT). The next level of classification distinguishes between information regarding characteristics within the robot system from those of the environment (INTERNAL vs. EXTERNAL). The classification was refined one more step to differentiate information relating GLOBAL events and characteristics from those that are purely LOCAL. The final level distinguishes information regarding the nonchanging parameters of the environment or robot-object relationship (STATIC) from information pertaining to the changing nature of the world (DYNAMIC).

This classification scheme, as with any scheme, achieves its usefulness from the fact that it allows one to organize thoughts regarding sensory information and provide some access, when viewed in light of a sensor taxonomy, to a mapping between a desired piece of information and a sensor or sensor technology that may provide that information. The next section develops a taxonomy for robot sensors.

### Taxonomy of Robot Sensors

In the robot domain a sensor is a device which transduces a physical quantity into a form amenable to utilization by a machine. Several authors have presented discussions regarding the sensors available to machines (1-4) and have proposed various classification schemes. A reasonable approach to a discussion of sensors is to delineate the various classes of transducers. The class structure presented here is based on the quantities transduced by the sensors and the manners in which the sensors interact with the environment. It is similar to that in Ref. 3, but inverted.

Robot senses provide information regarding internal quantities such as joint angles and motor currents as well as external quantities, e.g., through vision, touch, and rangefinding. However, with the one notable exception of wrist-force-sens-

ing research, the term "robot sensors" is typically thought to include only those devices that are local to the robot, i.e., those that provide DIRECT information, and devices that transduce external events, i.e., those that yield EXTERNAL information. Note that a number of different sensors may be used to determine a particular piece of information; for instance, vision can be used to deduce joint position.

Sensors are categorized depending on a number of factors (see Fig. 2). The highest level of the classification discussed below involves the manner in which the sensor transduces the target quantity and includes ACTIVE, PASSIVE, SAMPLE, and INVESTIGATIVE. In the next level sensors are categorized depending on how "intimate" the sensing device must be with the object being sensed or physical quantity being transduced.

**ACTIVE, PASSIVE, SAMPLE, INVESTIGATIVE.** A sensor is termed PASSIVE if it relies only on energy from the object or environment in order to transduce the quantity of interest. In this case the energy must be reflected, transmitted, or emitted by the object. This is in contrast to an ACTIVE sensor, which emits energy into the environment and compares the returned signal with the emitted signal to obtain an indication of the physical quantity of interest. Both PASSIVE and, in the ideal case, ACTIVE sensors leave all portions of the physical environment undisturbed. A SAMPLE sensor varies from the two above in that it physically captures a small portion of the environment and analyzes it for the feature(s) of interest. An olfactory sensor would analyze a sample of the atmosphere to provide a description of the "smell." A final class may be termed INVESTIGATIVE and encompasses the case in which the transducer must be extended into the environment in the form of a probe. This is meant to include cases such as a tactile sensor used to explore the immediate environment and not instances where, e.g., a tube is extended to collect an uncontaminated sample of the atmosphere. The ordering given above of PASSIVE, ACTIVE, SAMPLE, INVESTIGATIVE is in increasing interaction with the environment. In the case of a PASSIVE sensor the transducer essentially only monitors the environment, whereas an INVESTIGATIVE sensor interacts with the environment in a dynamic fashion. It is important to note that in some cases a particular sensor may fall into one category when used in one fashion and into another category when used differently. For example, a tactile array sensor—when used to cover the work space—is a PASSIVE device in that the stimulus object's energy and movement cause interaction with the device. When the sensor is attached to the fingertip of a robot's gripper and used to probe the environment, it falls into the INVESTIGATIVE category.

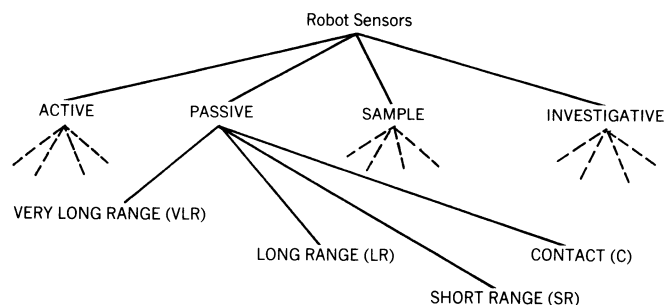


Figure 2. Taxonomy of robot senses.

**VLR, LR, SR, C.** The second distinguishing characteristic relates the proximity of the sensor to the stimulus being measured. At one end of the spectrum are those devices that must be in direct contact with the object exhibiting the characteristic to be transduced. These devices are, obviously, termed **CONTACT**, or **C**, devices. The other extreme includes the set of devices that transduce the phenomenon of interest from a great distance and are termed **VERY LONG RANGE**, or **VLR**, devices. Between these two classes are two rather arbitrarily defined classes. The class closest to the **CONTACT** end of the spectrum contains sensors that work in close proximity to, but not in contact with, the object being sensed. This class is termed **SHORT RANGE**, or **SR**. The final class, **LONG RANGE**, includes those devices that operate at distances greater than those of the **SR** sensors, i.e., greater than a few meters, and at distances shorter than the **VLR** devices, i.e., within a few kilometers.

### Robot Sensors

Various sensors are outlined below with the discussion generally following the structure of the taxonomy as presented in Figure 2. Within these major classes, sensors in each of the subclasses are discussed.

#### Vision

**Static Vision.** The most prominent sense to the majority of humans and the area receiving the greatest attention in the past is that of vision. Here vision is defined as the acquisition and processing of two-dimensional (possibly time-varying) arrays of numbers representing the light intensities of a scene in the real world. The number in element of the picture, called a picture element or "pixel," is produced by converting the amount of light sensed at that point in the image to a digital number. This process is known as digitization. Figure 3 contains a digitized image of a "T" plumbing connector.

The basic paradigm for using vision for robot control involves taking a "snapshot" of the scene currently in the view of the camera. This image is then processed to yield a description of the objects in the scene and their relationships to one another. Under this paradigm, the camera is termed **PASSIVE** since it utilizes electromagnetic radiation solely from the environment. In addition, it may belong to the **VLR**, **LR**, or **SR** classes depending on the manner in which it is used. During the past several years a great deal of effort has been directed toward the problems involved with static scene analysis as evidenced by the work described in Ref. 5-10 as well as at the University of Massachusetts (11-13), the Massachusetts Institute of Technology (14), at Maryland (15), at SRI International (16,17), at Stanford (18), at Carnegie-Mellon University (19,20), and a large number of other researches—a good sampling of this work is provided in *Computer Vision Systems*, edited by Hanson and Riseman (8); see also Refs. 21 and 22.

Machine-vision research has focused on several specific application domains: remote sensing data (typically from satellites), outdoor scenes (including military applications), biomedical image processing, and limited industrial settings. The applications of machine-vision research and techniques to industrial environments are, however, continually increasing in both number and sophistication. In most robotic applications the image of the world is typically thresholded to produce a binary image in order to reduce the storage required for the

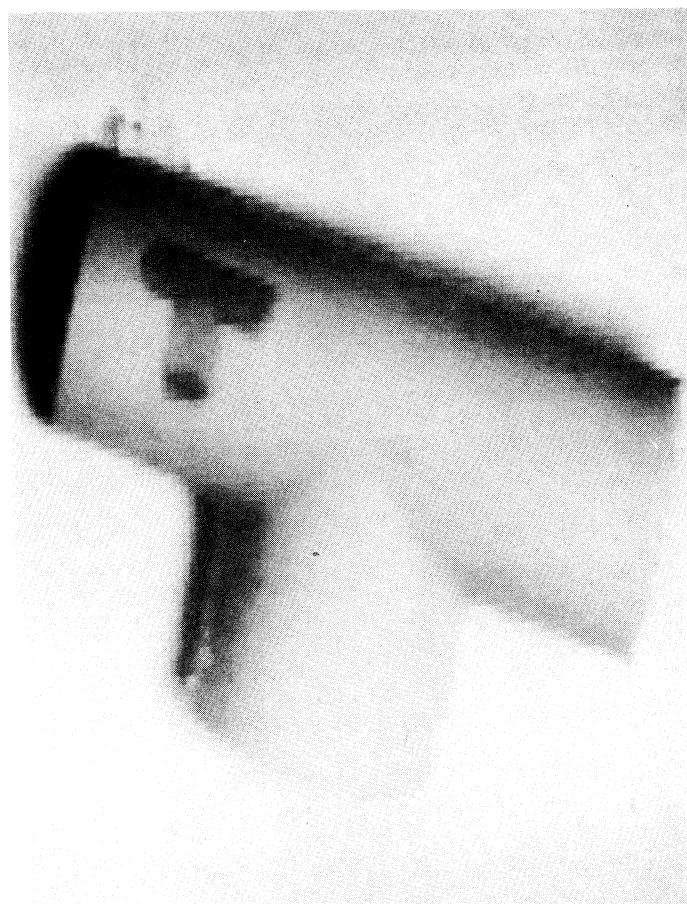


Figure 3. Digitized image of T plumbing connector.

image and the amount of computation required during the succeeding processing stages. Due to the relevance of vision to the subject of this work, only a brief discussion of current applied vision research efforts is appropriate. For further discussions of vision research applied to the robot domain, see Refs. 2 and 23-38.

Although the majority of machine-vision work has focused on military domains and outdoor scenes, there is a growing number of industrial applications in which some degree of vision is employed. Most of the work applied to robotics has been limited to the visible spectrum; however, research is in progress that utilizes alternative spectra, e.g., infrared, X-ray, etc. The current trend in most industrial robot-vision systems consists of calculating statistics for any dark objects found on a light background (or vice versa) (39). The Automatix system is an exception in that it is designed to handle both multiple thresholds and some gray-level processing (40). From these feature vectors the likely identity of the part can be chosen from a group of pre-specified parts. In the completely static case the orientation is determined from the feature values and the model of the part, thus allowing a robot arm to obtain the part.

In the case of system such as **CONSIGHT** (41-43) (see also Ref. 44) for a discussion of a similar system), a single binary image is constructed by a linear camera as the object passes by on a conveyer. A light stripe projector is employed by the camera's system to illuminate the scene. Since the system emits energy into the environment, this configuration is termed **AC-**

TIVE. The domain of this system is the industrial work place and the nature of this domain coupled with the effective range of the light projector makes this system fall into the LR class. The image is processed to locate the position, as defined by a particular feature of the object, and the orientation. The speed of the conveyer is monitored, and the location of a part can be predicted from the speed and processed image. Thus a manipulator can track and intercept an object.

**Motion Vision.** The examples just given involve the processing of static images. Motion vision, i.e., sequences of static images, is beginning to receive attention (45–50) but has yet to be utilized in the robotics domain to any extent. Dynamic image processing involves the collection and analysis of a sequence of images, usually from the same camera, where the images in the sequence are separated by equal, known, time intervals. Such techniques (see Refs. 45 and 50) can be used to extract this range information as well. The technique known as optic flow (48,51–54) also has great potential in this respect. Again, the techniques just mentioned are PASSIVE. The previous discussion assumed that the camera was fixed relative to the robot. An interesting alternative paradigm involves moving the camera in a specified manner and using knowledge about the motion and the resulting image data to extract information about the robot's spatial relationship with the environment, i.e., placing a camera in the robot gripper. In such a case the sense may be termed INVESTIGATIVE.

Thus, static robot vision, as typified by the utilization of binary or gray scale images from optical cameras, is usually PASSIVE and provides a 2-D image of the 3-D world as variations in the sensed intensity of energies in a particular portion of the electromagnetic spectrum. However, that machine vision can be ACTIVE as described above. Vision is typically thought of as applying to objects and scenes on a size scale comparable to scenes encountered in everyday life. However, machine vision also encompasses the world as seen through a microscope or from a satellite. Thus, depending on the situation, vision may be termed VLR, LR, or SR.

**Range Sensing.** Vision as described above lacks explicit information regarding the third dimension. The images are processed to extract information about the scenes. Rangefinding sensors are utilized to provide this information directly (55) where the information from the sensor indicates the distance from the sensor to points in the world. Acoustic or laser techniques are typically used to construct depth images of the environment where energy is emitted into the environment and its reflection sensed. Thus these devices are ACTIVE. As in the case of the CONSIGHT system, the effective range of these systems is limited, thus making them fall into the LR and SR classes. Alternatively, stereo vision can be used to derive the depths via triangulation. In this technique two static images of the same scene are collected from slightly different positions (55–57). The location in each image of a particular feature and the known parameters of the optical system are used to calculate the distance of the feature. These systems are PASSIVE and tend to be either VLR or LR.

There are several rangefinding techniques that are ACTIVE. In laser rangefinding a laser beam is used to scan the environment. The reflected light is sensed by the system, and the time of flight is used to determine the distance from the sensor to the reflecting object. An alternative approach involves triangulation wherein the laser beam is used to illuminate a point in the environment. The geometric parameters of the laser and the optical sensor are used to calculate the 3-D

coordinates of the point. Examples of rangefinding systems and techniques are provided in Refs. 58–64. These systems can produce a 2-D image of the world, a range map, where an element in the picture, a "rangel," is indicative of the distance to the nearest object in the environment along a line from the point in the sensor plane. Figure 4 shows a surface plot of a range image of the T connector shown in Figure 3. The value of each rangel in the image is displayed as a height above the horizontal plane. Since rangels encode the locations in space of points on the surfaces of objects, the data can be rotated and scaled to allow viewing from any viewpoint.

Acoustic imaging functions in a similar manner. In the simplest case a set of pulses at varying frequencies is emitted into the atmosphere. The sensor monitors the echo and measures the time of flight for the peak of the echo. This provides the distance to the largest near object. Radar and sonar systems are extensions of this simple paradigm in which the direction of the returning signal is monitored to provide a distance map similar to the laser technique. In the cases mentioned above the sensor emits energy into the environment and monitors the reflected energy to determine distance.

There have been efforts directed toward marrying the vision work with the ACTIVE approach for the rangefinding techniques. As discussed above, typical vision systems rely on ambient lighting. Here the illumination is controlled by the sensing system. Structured light is projected into the environment, a picture is taken, and the warped pattern of light is indicative of the spatial characteristics of the objects in the environment. Albus, among others, uses this approach (65), and the CONSIGHT system (41–43) can be viewed as functioning in this paradigm.

The vision and rangefinding techniques are used to provide information regarding the machine's immediate surroundings. For example, Bolles and Fischler used range data to locate cylinders in images of jumbled industrial parts (66) (see also Stereo vision). Both can be used to provide detailed information about a localized area (foveation) as well as less information about a much wider view. The interest in robotics is

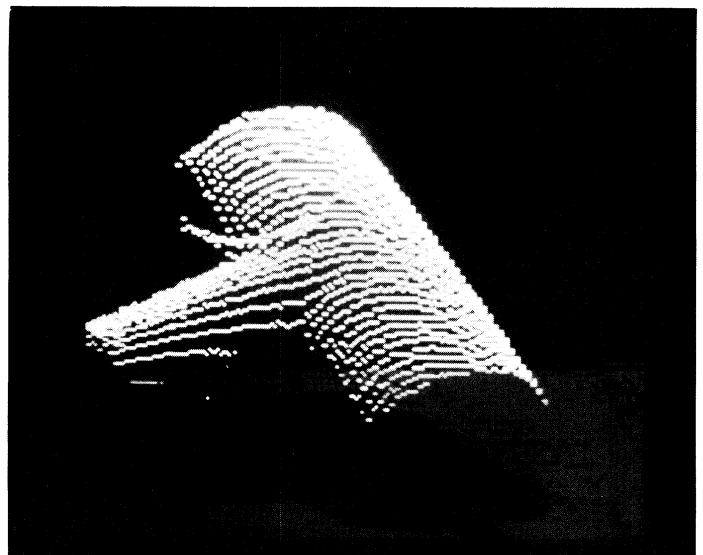


Figure 4. Surface plot of range image of T connector in Figure 3.



not necessarily in measuring the distance to the part but in understanding the identity and orientation of the parts (and features of parts) whose sizes are of the same order as the size of the robot.

**Proximity Sensing.** Noncontact sensing techniques have also been developed for use in determining short distances (67,68). This sensing is known as proximity sensing and can be either active or passive (see Ref. 69 for an in-depth review of this area). One approach to ACTIVE proximity sensing consists of an LED/phototransistor pair (39,70–73). This can be extended to an array of detectors as in the work of Okada (74). Typically, an LED emits light into the environment, and the phototransistor receives light reflected from objects. Crosnier used this approach for sensors inside the fingers of a gripper to detect movement of the surface due to contact with an object (75). In another approach a jet of air is blown into the environment and disturbance of the flow is monitored (76). In all cases the sensor provides distance information for an object near to the sensor. The use of an electromagnetic field to transduce eddy currents in nearby conductive plates is an example of a PASSIVE proximity sensor. All of these proximity sensors work on a distance scale that is much smaller than that generally used in vision and provide information only when an object is very near to the transducer.

The sensors just described all fall into the general calls of NONCONTACT sensors, which includes the classes of VLR, LR, and SR sensors. The remainder of the sensors to be considered are of the CONTACT type.

**Contact Sensing.** The most immediate aspect of the world surrounding the robot is the atmosphere. Transducers of the temperature, density, moisture, content, and composition of the atmosphere are all contact sensors. A temperature sensor can be PASSIVE in nature, i.e., a thermometer, or ACTIVE, i.e., a thermal conductivity sensor. The latter provides information useful in the analysis of the composition of objects with which the robot is in contact. Sensors transducing the density, moisture content, and composition of a gas may be classified as SAMPLE sensors since a small portion of the atmosphere is captured and analyzed.

Aside from the atmosphere, contact sensors are used to provide information regarding objects in the immediate vicinity of the machine. Proximity sensors have already been mentioned in the PASSIVE and ACTIVE cases. A Whisker sensor is a proximity transducer of the INVESTIGATIVE type. Here a fine, flexible shaft acts as a probe. When the probe contacts an object, the shaft bends and this information is available to the robot (77).

**Tactile Sensing.** Once in contact with an object, tactile sensation comes into play. There are three general types of information of interest: the physical parameters regarding the composition of the object; the parameters describing the interaction with the object, e.g., the forces and torques; and the object's shape. The former class includes such quantities as thermal conductivity, hardness, moisture content (useful in understanding slip), mechanical resonance properties, eddy-current transduction, mass, and location of centroid. Of these properties, only the latter two are unavailable from a purely local sensor. The mass and centroid can be derived from internal sensors.

The desired information regarding the interaction between

the robot and an object can vary in complexity from simple contact detection to force and torque parameters. In the simplest case interaction between the robot and its environment takes the form of the robot bumping into an object. A contact switch located on the surface of the robot can be used to determine collision with an object contacting the robot at that point. Knowledge of contact may be sufficient in many instances; yet in others the direction and magnitude of the contacting force may be needed. Matsushima et al. have experimented with an instrumented annular bumper with which the direction and magnitude of contact force can be determined (78). The bumper appears as an elastically attached skirt around the robot. The resting position of the center of the bumper is known and compared with the new location after contact. Knowledge of the displacement of the center, points of attachment of the bumper to the robot, and the spring constants of the supporting members allows computation of the force at the point of contact. Figure 5 is a photograph of a robot gripper and wrist used for research in robot sensing. It contains force sensors in the wrist and fingers, position and force sensing in the mechanism actuating the fingers, and a tactile array sensor mounted on each fingertip.

**Force Sensing.** The interface between the robot gripper and the manipulanda is characterized by the forces being applied by the fingers, slip of the object relative to the gripper, and the forces present at distant points in the arm resulting from manipulating the object. Yashikata reported on a gripper with strain-gauge force sensors in the fingers which could be used to direct grasping (79). Another facet of this information involves the principal forces (80) present at the gripper. These include the translational forces along, and the rotational forces about, and  $x$ ,  $y$ , and  $z$  axes. A typical approach to measuring these forces is to construct a sensing element that fits between the gripper and the wrist of the robot or between a fixture and the work table (80,81). Specific examples of such devices are the instrumented remote center compliance device, or IRCC, i.e., the wrist developed by Seltzer (82), and the Force Sensing Wrist developed by Lord Corporation. These devices are instrumented with strain gauges that provide information regarding the deformation of the structure. Proper design of the unit allows determination of all six force vectors. Such devices have been used to successfully guide the mating of close-tolerance parts (83–85). The subject of slip has been the target of several ingenious mechanical sensors (86,87) wherein instrumented needles, wheels, or tracks in the fingers are used to

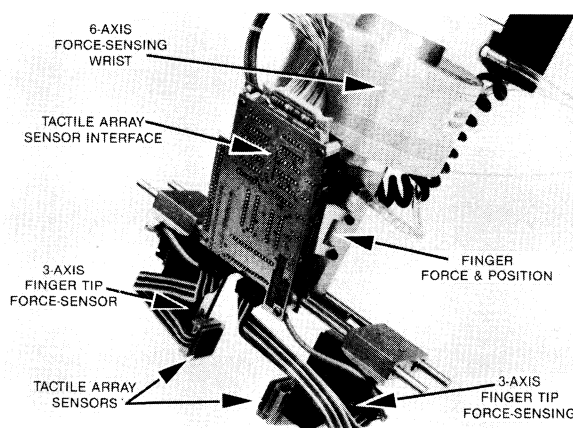


Figure 5. Robot gripper used for research in robot sensory systems.

detect movement of the object relative to the gripper but nothing is generally available. The magnetoelastic-array-sensor approach discussed by Hackwood (88) may lead to a device capable of providing an "image" of the forces tangential to the sensor pad.

The forces present on the individual fingers of a gripper can also be determined. The IBM RS1 manipulator can be configured with a two-fingered, parallel-jaw gripper where the fingers are instrumented with strain gauges so that the pinching, side, and end forces can be transduced (89). The gripper need not have "fingers" that resemble those of humans. In work done by Trevelyan (90,91) in Australia, the end effector assumes the form of a wool-shearing tool (see Ref. 92). The force being applied by the tool is used to modify the trajectory of the tool during the shearing process.

**Tactile Sensors.** The final class of tactile sensors can be divided into three general categories depending on the type of information they provide. Contact sensors compose the first class and yield information indicating either the presence or absence of physical stimuli. A robot hand with contact sensing on several sides was developed by Goto (93) to handle blocks placed on a table. A high spatial resolution conductive rubber contact sensing array has been developed at the Artificial Intelligence Laboratory at MIT and has been successfully used to identify parts from a small class learned by the system (94). An interesting use of contact sensing has been reported by Hirose and Umetani (95). Here, simple on-off, or "binary touch," sensing on each segment of an active cord, or snake, mechanism was used to guide the motion of the device.

The second category of tactile sensors contains the sensors that provide information regarding the magnitude of the force at each sensing point (see Ref. 19 for a review of some of the recent sensor designs). These devices produce an "image," called a force image, of the forces present at the interface between the sensor and the object. The magnitude of each element in the image, called a "forcel," represents the force sensed at the corresponding point on the sensor surface. Figure 6 contains a force image from a small metal washer.

A hand constructed by Hill and Sword (96) employed analog force-sensing arrays. Recent sensors have utilized woven graphite fibers as the physical transducer (97,98). Another approach utilizes the Hall effect to transduce the positions of a set of tactile probes (99). Hall cells are moved by contact with the stimulus object, thus changing the distance between the cells and magnets fixed to the gripper. This change is measured via the Hall cells and made available to the controlling computer. The "artificial skin" tactile-array sensor developed by Clot and Briot at the Laboratory for Automation and Systems Analysis in Toulouse utilizes pressure-variable transverse resistance to transduce the forces present on the pad (100-101). The device used in robot experiments consist of 100 points arranged in a  $10 \times 10$  array with 1-cm spacing between the centers of adjacent transducers and a printed circuit board on which is located an array of "sensitive points." These transducers consist of a center-measurement electrode surrounded by an annular electrode. A homogeneous pad composed of a conductive elastomer 5 mm thick is placed on top of the contact array. A reference voltage is applied to the annular electrode, and the current entering the center electrode is measured. Since this current is proportional to the resistance through the pad, it is an indication of the force present in that area. This sensor is very similar in design to the device developed at JPL (73,103).

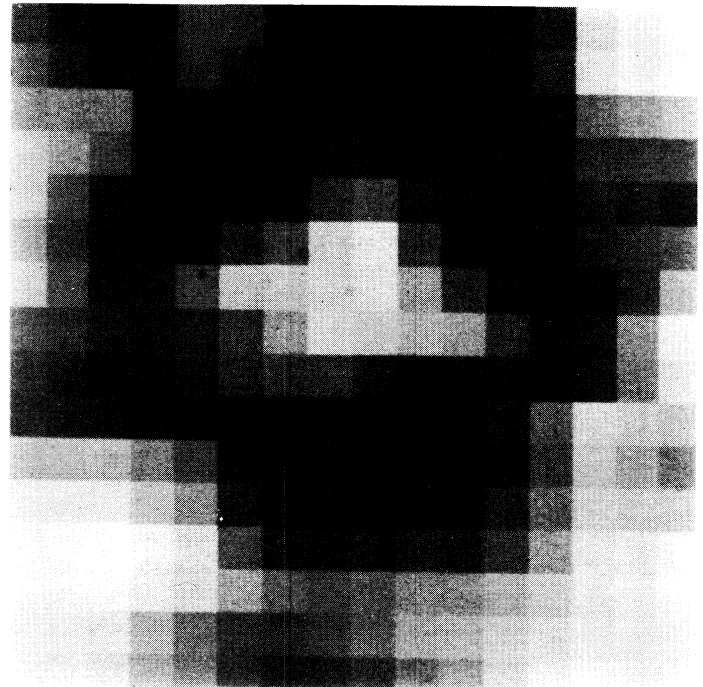


Figure 6. Force image of a small washer.

Recent work by Raibert and Tanner (104) has been directed toward the development of a tactile-array sensor employing VLSI technology. The intent is to place processing capabilities at each forcel (force element) in the array, thus distributing the computation. In another recent effort (105,106) piezoelectric elements are used as the basic transduction device. The elements are placed in opposition to one another and separated by a compliant filler. The element closest to the surface of the sensor is used to produce an acoustic wave in the filler that is received by the second element. The time of flight is proportional to the separation of the elements, which in turn is proportional to the force applied by the stimulus since force leads to compression of the filler.

The final class of tactile-array sensors includes sensors designed to transduce the 3-D shape of an object. This is similar to range sensing except that the tactile sensor is a CONTACT device and provides only LOCAL information where range sensors are NONCONTACT devices providing GLOBAL information. In one example application a linear array of spring-mounted probes was used to return the contour of the surface. Probe height was transduced by strain gauges mounted on the springs. Page and Pugh (107) presented a sensor that builds a contour map of the object being sensed as the sensor is lowered onto the object. Takeda constructed a hand with arrays of free-floating needles as sensors. The needles would conform to the shape of the surface being touched, thereby providing identification information (108).

## BIBLIOGRAPHY

1. A. J. Barbera, J. S. Albus, and M. L. Fitzgerald, Hierarchical Control of Robots Using Microcomputers, *Proceedings of the Ninth International Symposium on Industrial Robots*, Washington, DC, March 13-15, 1979.
2. C. A. Rosen, Machine Vision and Robotics: Industrial Requirements, SRI International Technical Note 174, November 1978.

3. D. Nitzan, Assessment of Robotic Sensors, paper presented at NSF Robotics Research Workshop, Newport, RI, April 15-17, 1980.
4. D. Nitzan and C. A. Rosen, "Programmable industrial automation," *IEEE Trans. Comput.* C-25(12), 1259-1269 (1976).
5. J. B. Burns, A. R. Hanson, and E. M. Riseman, Extracting Linear Features, *Proceedings of the International Conference on Pattern Recognition*, Montreal, Canada, August, 1984.
6. P. A. Nagin, A. R. Hanson, and E. M. Riseman, "Studies in global and local histogram-guided relaxation algorithms," *IEEE Trans. Patt. Anal. Mach. Intell.* PAMI-4(3) (May 1982).
7. A. R. Hanson, E. M. Riseman, and F. C. Glazer, Edge Relaxation and Boundary Continuity, in R. Haralick (ed.), *Consistent Labeling Problems in Pattern Recognition*, Plenum, New York, 1980.
8. A. R. Hanson and E. M. Riseman (eds.), *Computer Vision Systems*, Academic Press, New York, 1987.
9. A. R. Hanson and E. M. Riseman, Segmentation of Natural Scenes, in A. R. Hanson and E. M. Riseman (eds.), *Computer Vision Systems*, Academic Press, New York, pp. 129-163, 1978.
10. B. W. York, A. R. Hanson, and E. M. Riseman, A Surface Representation for Computer Vision, *Proceedings of the IEEE Workshop on Picture Data Description and Management*, Asilomar, CA, August 1980.
11. B. K. P. Horn, "Understanding image intensities," *Artif. Intell.* 8, 201-231 (1977).
12. B. K. P. Horn, Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View, in P. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.
13. D. Marr, "Early processing of visual information," *Philos. Trans. Roy. Soc.* B275, 483-524 (1976).
14. D. Marr, Representing Visual Information, in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman (eds.), Academic Press, New York, pp. 61-80, 1978.
15. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, New York, 1976.
16. J. M. Tenenbaum and H. G. Barrow, "Experiments in interpretation-guided segmentation," *Artif. Intell.* 8(3), 241-274 (1977).
17. H. G. Barrow and J. M. Tenenbaum, Recovering Intrinsic Scene Characteristics from Images, in A. R. Hanson and E. M. Riseman (eds.), *Computer Vision Systems*, Academic Press, New York, pp. 3-26, 1978.
18. T. O. Binford, R. A. Brooks, and D. G. Lowe, Image Understanding Via Geometric Models, *Proceedings of the Fifth International Conference on Pattern Recognition*, Miami, FL, December 1980, pp. 364-369.
19. "Progress in tactile sensor development," *Robot. Tod.* 5(3) (June 1983).
20. T. Kanade, Recovery of the Three-Dimensional Shape of an Object from a Single View, CMU U-CS-79-153, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, October 1979.
21. D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
22. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
23. A. Pugh, *Robot Vision*, Springer-Verlag, New York, 1983.
24. G. J. Agin, An Experimental Vision System for Industrial Application, *Proceedings of the Fifth International Symposium on Industrial Robots*, Chicago, IL, 1975.
25. G. J. Agin and R. O. Duda, SRI Vision Research for Advanced Industrial Automation, *Proceedings of the Second USA-Japan Computer Conference*, 1975, pp. 113-117.
26. G. J. Agin, Real Time Control of a Robot with a Hand-Held Camera, *Proceedings of the Ninth International Symposium and Exposition on Industrial Robots*, Washington, DC, March 1979, pp. 233-246.
27. K. Armbruster et al., A Very Fast Vision System for Recognizing Parts and Their Location and Orientation, *Proceedings of the Ninth International Symposium and Exposition on Industrial Robots*, Washington, DC, March 1979, pp. 265-280.
28. G. J. Gleason and G. J. Agin, A Modular Vision System for Sensor-Controlled Manipulation and Inspection, *Proceedings of the Ninth International Symposium and Exposition on Industrial Robots*, Washington, DC, March 1979, pp. 57-70.
29. R. C. Bolles, Part Acquisition Using the SRI Vision Module, *Third International Computer Software and Applications Conference*, November 5-8, 1979, IEEE Computer Society, Chicago, IL.
30. T. Vámos, M. Bathor, and L. Mero, A Knowledge-Based Interactive Robot-Vision System, *Proc. of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, 1979, pp. 920-922.
31. W. A. Perkins, Model-Based Vision System for Scenes Containing Multiple Parts, *Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, August 1977, pp. 678-684. Also available as GM Research Lab Report GMR 2386, 1977.
32. W. A. Perkins, "A model-based vision system for industrial parts," *IEEE Trans. Comput.* C-27, 126-143 (February 1978).
33. M. Yashida and S. Tsuji, A Machine Vision for Complex Industrial Parts with Learning Capacity, *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, 1975, pp. 819-826.
34. S. W. Olsztyn et al., An Application of Computer Vision to a Simulated Assembly Task, *Proceedings of the First International Joint Conference on Pattern Recognition*, 1973, pp. 505-513.
35. W. B. Heigenbotham et al., Visual Feedback Applied to Programmable Assembly Machines, *Proceedings Second International Symposium on Industrial Robots*, May 1972, pp. 77-88.
36. H. Toda and I. Masaki, Kawasaki Vision System: Model 79A, Kawasaki Heavy Industries, Ltd., Robot Section, Nishi-Kobe Works, Matsumoto, Hazentani-Cho, Japan.
37. J. R. Birk et al., General Methods to Enable Robots with Vision to Acquire, Orient, and Transport Workpieces, Sixth Report to NSF, August 1980.
38. R. N. Nagel, G. J. VanderBrug, J. S. Albus, and E. Lowenfeld, Experiments in Part Acquisition Using Robot Vision, SME Technical Paper MS79-784, Society of Manufacturing Engineers, Dearborn, MI, 1979.
39. J. Y. Catros, "Use of optical reflectance sensors in robotics applications," *IEEE Trans. Sys. Man Cybernet.* SMC-10(12) (December 1980).
40. A. G. Reinhold and G. Vanderbrug, "Robot vision for industry: The autovision system," *Robot. Age*, 22-28 (Fall 1980).
41. M. R. Ward, L. Rossol, S. W. Holland, and R. Dewar, CONSIGHT: A Practical Vision-Based Robot Guidance System, General Motors Research Laboratories Publication GMR-2912, February 1979.
42. S. W. Holland, J. T. Olsztyn, L. Rossol, and R. Dewar, A General-Purpose Visual Object Locator, General Motors Research Laboratory Report CS-181, April 1976.
43. S. Holland, L. Rossol, and M. Ward, CONSIGHT-I: A Vision Controlled Robot System for Transferring Parts from Belt Conveyors, General Motors Research Symposium "Computer Vision and Sensor-Based Robots," September 1978.
44. R. J. Popplestone et al., Forming Models of Plane and Cylinder Faceted Bodies from Light Stripes, *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tbilisi, Georgia, September 1975, pp. 664-668.
45. T. D. Williams, Computer Interpretation of a Dynamic Image from a Moving Vehicle, Ph.D. Dissertation, Computer and Infor-

- mation Science Department, University of Massachusetts, Amherst, MA, May 1981.
46. T. D. Williams, "Depth from camera motion in a real world scene," *IEEE Trans. PAMI* **PAMI-2**(6), 511-516 (Nov. 1980).
  47. D. T. Lawton, Processing Dynamic Images from a Moving Sensor, Ph.D. Dissertation, Computer and Information Science Dpt., University of Massachusetts, Amherst, MA, Feb. 1984.
  48. D. T. Lawton, Optic Flow Field Motion and Processing Image Motion, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., Aug. 1981.
  49. D. T. Lawton, Constraint-Based Inference for Optic Flow, *Proceedings of the First National Conference on Artificial Intelligence*, Stanford University, August 18-21, 1980.
  50. J. M. Prager and M. A. Arbib, "Computing optic flow: The MATCH algorithm and predictions," *Comput. Vis. Graph. Im. Proc.* **24**, 271-304 (1983).
  51. J. J. Gibson, *The Perception of the Visual World*, Houghton Mifflin, Boston, MA, 1950.
  52. F. Glazer, Computing Optic Flow, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., Canada, August 1981.
  53. D. N. Lee and J. R. Lishman, "Visual control of locomotion," *Scand. J. Psychol.* **18**, 224-230 (1977).
  54. K. Prazdny, Motion and Structure from Optical Flow, *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, August, 1979.
  55. A. C. Kak, Depth Perception for Robots, S. Nof (ed.), *Handbook of Industrial Robotics*, Wiley, New York, 1985; also Purdue University School of Electrical Engineering Report TR-EE 83-44, October 1983.
  56. D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science* **194**, 283-287 (October 1976).
  57. J. R. Woodham, Photometric Stereo: A Reflectance Map Technique for Determining Surface Orientation from Image Intensity, *Proceedings of the Twenty Second SPIE Technical Symposium*, San Diego, CA, August 1978.
  58. F. J. Pipitone and T. G. Marshal, "A wide-field scanning triangulation rangefinder for machine vision," *Robot. Res.* **2**(1) (Spring 1983).
  59. D. Nitzan, A. E. Brain, and R. O. Duda, "The measurement and use of registered reflections and range data in scene analysis," *Proc. IEEE* **65**, 206-220 (February 1977).
  60. D. Nitzan, Scene Analysis Using Range Data, Technical Note 69, Artificial Intelligence Center, SRI International, August 1972.
  61. K. Sugihara, Automatic Construction of Junction Dictionaries and Their Exploitation for the Analysis of Range Data, *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, 1979.
  62. A. R. Johnston, Infrared Laser Rangefinder, NASA New Technology Report No. NPO-13460, Jet Propulsion Laboratory, August 1973.
  63. Y. Shirai and M. Suwa, Recognition of Polyhedrons with a Rangefinder, *Proceedings of the Second International Joint Conference on Artificial Intelligence*, London, September 1971, pp. 80-87.
  64. Y. Nishimoto and Y. Shirai, "A Parallel Matching Algorithm for Stereo Vision," *Proc. of the Ninth IJCAI*, Los Angeles, CA, 1985, pp. 977-980.
  65. G. J. VanderBurg, J. S. Albus, and E. Barkmeyer, A Vision System for Real Time Control of Robots, *Proceedings of the Ninth International Symposium and Exposition on Industrial Robots*, Washington, DC, March 1979, pp. 213-323.
  66. R. C. Bolles and M. A. Fischler, A RANSDAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., August, 1981, pp. 637-643.
  67. T. Okada, "A short-range finding sensor for manipulators," *Bull. Electrotech. Lab.* **42**(6) (1978).
  68. A. R. Johnston, Proximity Sensor Technology for Manipulator End Effectors, *Proceedings of the 2nd Conference on Remotely Manned Systems*, California Institute of Technology, Pasadena, CA, 1975.
  69. W. D. Koenigsberg, Noncontact Distance Sensing Technology, Presented at Seminar and Workshop on Sensors for Robotics and Automation, Kingston, RI, June 8-9, 1983.
  70. J. Y. Catros et al., Automatic Grasping Using Infrared Sensors, *Proceedings of the Eighth International Symposium on Industrial Robots and Fourth International Conference on Robot Technology*, Stuttgart, FRG, May 30-June 1, 1978.
  71. A. K. Bejczy, Smart Sensors for Smart Hands, *Proceedings of the AIAA/NASA Conference on "Smart" Sensors*, Hampton, VA, 1978.
  72. A. K. Bejczy, "Effect of hand-based sensors on manipulator control performance," *Mech. Mach. Th.* **12** (1977).
  73. B. Espian and J. Y. Catros, "Use of optical reflectance sensors in robotics applications," *IEEE Trans. Sys. Man Cybernet.* **SMC-10**(12) (December 1980).
  74. T. Okada, "Development of an optical distance sensor for robots," *Robot. Res.* **1**(4) (Winter 1982).
  75. J. J. Crosnier, Grasping Systems with Tactil Sense Using Optical Fibres, in B. Rooks (ed.), *Developments in Robotics 1983*, IFS Publications, Kempston, Bedford, UK, 1983.
  76. H. Hanafusa and H. Asada, An Adaptive Control of Robot Hand Equipped with Pneumatic Proximity Sensory, *Proceedings of the Third Conference on Industrial Robot Technology and the Sixth International Symposium on Industrial Robots*, University of Nottingham, UK, March 24-26, 1976.
  77. G. A. Folchi et al., Computer Controlled Pneumatic Retractable Search Sensor, U.S. Patent 4,001,156, January 1977.
  78. J. G. Bollinger, "Using a tactile sensor to guide a robotic welding machine," *Sens. Rev.* (March 1981).
  79. K. Yoshikata, A Microcomputer Controlled Tactile Gripper Sensor for Robot Manipulation and Effector Design, *Proceedings of the ASME 1983 International Computers in Engineering Conference and Exhibit*, Chicago, IL, August 1983.
  80. C. R. Flatau, Force Sensing in Robots and Manipulators, *Proceedings of the Second International CISM-IFTOMM Symposium on the Theory and Practice of Robots and Manipulators*, Warsaw, Poland, September 14-17, 1976.
  81. P. C. Watson and S. H. Drake, Pedestal and Wrist Force Sensors for Automatic Assembly, *Proceedings of the Fifth International Symposium on Industrial Robots*, Chicago, September 1975, pp. 501-511.
  82. D. S. Seltzer, Use of Sensory Information for Improved Robot Learning, SME Technical Paper MS79-799, Society of Manufacturing Engineers, Dearborn, MI, 1979.
  83. T. Goto, K. Takfyasu, and T. Inoyama, "Control algorithm for precision insert operation robots," *IEEE Trans. Sys. Man Cybernet.* **SMC-10**(1) (January 1980).
  84. T. Goto, T. Inoyama, and K. Takeyasu, Precise Insert Operation by Tactile Controlled Robot HI-T-HAND Expert-2, *Proceedings of the Fourth International Symposium and Industrial Robots*, Tokyo, 1974, pp. 209-218.
  85. K. Takeyasu, T. Goto, and T. Inoyama, "Precision insertion control robot and its application," *Trans. ASME, J. Eng. Ind.* (November 1976).
  86. M. Ueda, K. Iwata, and J. Shingu, Tactile Sensors for and Industrial Robot to Detect Slip, *Proceedings of the Second International Symposium on Industrial Robots*, Chicago, IL, May, 1972.

87. L. D. Harmon, Touch Sensing Technology: A Review, presented at the Conference on Military and Space Applications of Robotics, held at the National Academy of Sciences, Washington, DC, November 3–5, 1980.
88. S. Hackwood, G. Beni, L. A. Hornak, R. Wolfe, and T. J. Nelson, "A torque-sensitive tactile array sensor," *Robot. Res.* 2(2) (Summer 1983).
89. *A Manufacturing Language Reference and Screen Editor*, IBM 7565 Manufacturing System Software Library, International Business Machines, Boca Raton, FL, 1982.
90. J. P. Trevelyan, "Software techniques for automated sheep shearing," *Aust. Comput. Bull.* (March 1982).
91. J. P. Trevelyan et al., Techniques for Surface Representation and Adaptation in Automated Sheep Shearing, *Proceedings of the Twelfth International Symposium on Industrial Robots*, Paris, France, June, 1982.
92. P. C. Wong and P. R. W. Hudson, The Australian Robotic Sheep Shearing Research Programme, *Proceedings of the Thirteenth International Symposium on Industrial Robots and Robot 7*, Chicago, IL, April 17–21, 1983.
93. T. Goto, Compact Packaging by Robot with Tactile Sensors, *Proceedings of the Second International Symposium on Industrial Robots*, Chicago, 1972.
94. W. D. Hillis, Active Touch Sensing, A.I. Memo 629, Artificial Intelligence Laboratory, MIT, Cambridge, MA, April 1981.
95. S. Hirose and Y. Umetani, Kinematic Control of Active Cord Mechanism with Tactile Sensors, *Proceedings of the Second International CISM-IFTOMM Symposium on the Theory and Practice of Robots and Manipulators*, Warsaw, Poland, September 14–17, 1976.
96. J. W. Hill and A. J. Sword, Manipulation Based on Sensor-Directed Control: An Integrated End Effector and Touch Sensing System, *Proceedings of the Seventeenth Annual Human Factor Society Convention*, Washington, DC, October 1979.
97. M. H. E. Larcombe, Carbon Fibre Tactile Sensors, *Proceedings of the First International Conference on Robot Vision and Sensory Controls*, Stratford-upon-Avon, UK, April 1981.
98. M. H. E. Larcombe, Tactile Sensors, Sonar Sensors, and Parallax Sensors for Robot Applications, *Proceedings of the Third Conference on Robot Technology and Sixth International Symposium on Industrial Robots*, University of Nottingham, UK, March 1976.
99. G. Kinoshita, S. Ohishi, and M. Yoshida, Development and Realization of a Multi-Purpose Tactile Sensing Robot, in B. Rooks (ed.), *Developments in Robotics 1983*, IFS Publications, Kempston, Bedford, UK, 1983.
100. J. Clot and J. Falipou, Realization D'otches Pneumatiques Modulaires, Etude D'un Detecteur de Pressions Plantaires, *Publication LAAS 1852*, Centre National de la Recherche Scientifique, Laboratoire d'Automatique et d'Analyse des Systems, Toulouse, France, December 1978.
101. J. Clot and J. Falipou, Project Pilote SPARTACUS: Etude d'un Capteur Tactile (Peau Artificielle) Utilise Comme Detecteur D'efforts de Pression et de Glissement, *Publication LAAS 1629*, Centre National de la Recherche Scientifique, Laboratoire d'Automatique et d'Analyse des Systems, Toulouse, France, December 1977.
102. M. Briot, The Utilization of an "Artificial Skin" Sensor for the Identification of Solid Objects, *Proceeding of the Ninth International Symposium and Exposition on Industrial Robots*, Washington, DC, March 1979, pp. 529–548.
103. W. P. Butler, Technical Support Package on Transducer with a Sense of Touch, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA, November 1979.
104. M. H. Raibert and J. E. Tanner, "Design and implementation of a VLSI tactile sensing computer," *Robot. Res.* 1(3) (1982).
105. R. Bardelli, P. Dario, D. DeRossi, and P. C. Pinotti, Piezo- and Pyroelectric Polymer Skin-Like Tactile Sensors for Robots and Prostheses, *Proceedings of the Thirteenth International Symposium on Industrial Robots and Robots 7*, Chicago, IL, April 1983.
106. P. Dario, C. Domenici, R. Bardelli, D. DeRossi, and P. C. Pinotti, Piezoelectric Polymers: New Sensor Materials for Robotic Applications, *Proceedings of the Thirteenth International Symposium on Industrial Robots, and Robots 7*, Chicago, IL, April 1983.
107. C. J. Page, A. Pugh, and W. B. Heginbotham, Novel Technique for Tactile Sensing in a Three-Dimensional Environment, *Proceedings of the Third Conference on Robot Technology and Sixth International Symposium on Industrial Robots*, University of Nottingham, March 1976.
108. S. Takeda, Study of Artificial Tactile Sensors for Shape Recognition—Algorithm for Tactile Data Input, *Proceedings of the Fourth International Symposium on Industrial Robots*, Tokyo, November, 1974, pp. 199–208.

K. OVERTON  
General Electric

## SHAKY

SHAKY is a robot developed at SRI. It operates in a "blocks-world" environment made up of uniformly colored prismatic solids. It uses a monocular-vision (qv) system based on edge detection (qv). SHAKY plans its path using a connected graph and the A\* algorithm (qv) (see H. P. Moravec, *Robot Rover Visual Navigation*, UMI Research, Ann Arbor, MI, 1981).

J. ROSENBERG  
SUNY at Buffalo

## SHAPE ANALYSIS

Shape analysis is a generic term used to identify methods that determine the form and the spatial arrangement of objects in the world. Vision (qv) and touch are the two sensory modalities used for shape analysis in human perception. Of the two, vision has received the most attention in AI. Several articles survey work in the field (1–3). Textbooks are available for further study (4–6). Shape analysis based on touch and active range sensing is an emerging area of importance in robotics (qv) (7).

This entry is about shape analysis in computational vision. Computational vision is the study of systems that produce descriptions of the world from images of that world. The purpose is to represent those aspects of the world that are required to carry out some task. For most tasks shape is a necessary component of the description produced. Here, the term "representation" is used to identify a formalism, or language, for encoding a general class of shapes. The term "description" is restricted to mean a specific expression in the formalism that identifies an instance of a particular shape or class of shapes in the representation.

In a general-purpose vision system the mapping from signal input to final shape description is too complex to be treated as a function in a single representation. Shape analysis requires many levels of intermediate representation. Identifying those levels and establishing the constraints that operate both within and between levels is the fundamental challenge of

computational vision research. Each level of representation must consider both the processes that derive the representation and the processes that compute with the representation. At the level of the signal one deals with descriptions that can be derived directly from the image. This leads initially to representations for the 2-D shape of image patterns. Interpreting image properties as scene properties leads to representations for the visible surfaces in the scene. Finally, recognition of distinct objects and their spatial arrangement requires representations for 3-D shapes that are independent of viewpoint.

Over the past two decades there has been considerable growth in the theoretical base for computational vision. One major trend has been toward a concentration on topics corresponding to identifiable modules in human perception (8). This has led to the development of 3-D vision systems including shape from stereo (9,10), shape from contour (11–15), shape from motion (16) and optical flow (17), and shape from texture (11,18).

Computational vision distinguishes three levels of representation: 2-D shape, visible surfaces, and 3-D shape. Unfortunately, there is no general agreement on the right representations to use at each of these levels. There is some agreement on necessary criteria these shape representations must satisfy.

### Criteria for Shape Representation

Several authors suggest necessary criteria that general-purpose shape representations must satisfy (3,19–21). No single representation proposed to date satisfies all of the criteria. Nevertheless, the criteria provide a useful framework to discuss representations that have been proposed. The criteria are given below.

**Representation of Shape Must Be Computable Using Only Local Support.** The ability to derive the representation from the input data is the minimal requirement. Local support further stipulates that the representation can be computed locally. This is required to deal with occlusion and to perform detailed inspection. It is also of practical importance since processes that derive the representation can then be implemented efficiently.

**Representation of Shape Must Be Stable.** That is, small changes in the input should cause only small changes in the result. Images are subject to noise. Thus, stability is an important criterion for processes that derive initial descriptions from an image. Stability is also an important criterion for subsequent levels of representation because, without stability, it is difficult to define an effective measure of similarity to compare descriptions.

**Representation of Shape Must Be Rich in the Sense of Information Preserving.** Images are 2-D, while objects are 3-D. Image projection loses information. An image defines an equivalence class, usually infinite, of worlds that project to the identical image. A representation is rich if it does not arbitrarily restrict or extend this equivalence class. Rich representations are needed to describe a large class of objects, including objects that may never have been seen before.

**Representation Must Describe Shape at Multiple Scales.** Representations at multiple scales are useful for several reasons. First, representations at multiple scales suppress detail until it is required. Descriptions at a coarse scale relate to overall shape. Detail emerging at finer scales includes features that are more local. A pinhole in a metal casting is not significant when the task is to identify the part. But it is critical when the

task is to inspect the part for defects. Second, objects must be representable at different levels of detail. This can be accomplished using a hierarchical representation of shape that also takes into account the difference in object appearance owing to scale. For example, a forest is made up of individual trees. A forest can be represented hierarchically as a particular spatial arrangement and species composition of individual trees. At a coarser scale the forest must still be represented as a forest even when the individual trees are no longer discernible. Third, in the presence of noise there is an inherent trade-off between the detectability of an image feature and its precise localization in space. By working at multiple scales, it is possible to optimize this trade-off dynamically, as required. Fourth, a coarse-to-fine analysis can introduce significant computational speed-up in methods for shape analysis requiring search or convergence. Fifth, to be useful, a representation should be storage efficient. Representations at multiple scales are needed to be both storage efficient and rich.

**Representation Must Define Object-Based Semantics for Shape Description and Segmentation.** In general, comparison of 2-D shape descriptions fails because there is no stable similarity measure to use. Large changes in shape description follow from minor changes in either the spatial configuration of the objects in the world, the viewpoint, or the illumination. Shape analysis requires representations in which 3-D shape is explicit so that spatial relationships between surfaces can be computed easily. This is necessary to segment complex shapes into simpler components, to predict how objects will appear, and to deal with occlusion.

**Representation of Shape Must Correspond to Human Performance on Task.** Earlier, it was noted that an image defines an equivalence class of worlds that project to the identical image. Similarly, a representation defines equivalence classes of images that produce identical descriptions in the representation. Human perception also defines equivalence classes of worlds/images that produce identical perceptions. A representation of shape corresponds to human performance on some task if two conditions are satisfied. First, images that produce distinct descriptions in the representation are perceived as distinct in the task. Second, images that produce identical descriptions in the representation are perceived as identical in the task. A correspondence to human performance is difficult to achieve, in part because much remains to be understood about human perception. Nevertheless, developing this correspondence is a major motivating factor for current work in computational vision.

### Computational Task

The computational task is to determine the 3-D shape of objects from their 2-D projection onto images. The principal shape-analysis methods studied in computational vision are shape from contour, shape from stereo, shape from shading, shape from texture, and shape from motion. Although each method differs considerably in precise detail, all share a common characterization as computational tasks. The steps embodied in a shape analysis method are shown as follows.

**Identify Visual Task.** This involves picking a task domain and a class of locally computable image features for the domain that provide cues to 3-D shape.

**Derive Mathematical Equations That Describe How World Determines Image.** The equations are based on the laws of optics and, in general, consider both geometry and radiometry. The



equations determine the mapping from scene to image. Shape analysis, however, requires a solution to the inverse problem. That is, it must determine the mapping from image to scene.

**Demonstrate That Inverse Problem is Underconstrained.** It is usually straightforward to demonstrate that the problem is locally underconstrained. In general, the problem is also globally underconstrained although this can be more difficult to demonstrate.

**Identify Additional Constraints That Lead to Unique Solution to Inverse Problem.** Image features determine equivalence classes of possible scene features. Conceptually, a unique solution is obtained when a metric is applied to the equivalence classes to select a single preferred solution. Vision has been termed a conservative process (22). The metric is often expressed as a performance index designed to achieve smooth, regular, or minimal energy solutions. Identifying a suitable performance index is not a trivial matter. There are many possible measures to consider for a given visual task. Some degree of mathematical rigor is generally required to demonstrate that a particular choice does, in fact, lead to a unique solution. Finally, even when the existence of a unique solution is established, it is still necessary to develop an algorithm to determine the solution.

**Show That Solution Thus Obtained Agrees with Human Perception.** Whatever the metric, the correct physical solution cannot be obtained in all cases. Human perception does not always correspond to the correct physical solution either. One level of agreement with human perception is to demonstrate that the computed solution agrees with human perception for the chosen visual task. At a second level one also compares known algorithms for computing the solution to plausible mechanisms for biological implementation.

## Shape from Shading

A smooth opaque object produces an image with spatially varying brightness even if the object is illuminated evenly and is made of a material with uniform optical properties. Shading in an image provides essential information about object shape. Methods have been developed to determine shape from shading. These methods are based on an image irradiance equation formulated to determine image brightness as a function of surface orientation. The image irradiance equation cannot be directly inverted because surface orientation has two degrees of freedom and image brightness has only one. Additional information is required to reconstruct the visible surface. Many different constraints have been used in shape-from-shading methods. Here, two are considered. First, one can impose an overall smoothness metric on the desired solution using locations in the image where surface orientation is determined locally as initial conditions. Second, one can use multiple images in a technique known as photometric stereo. The development given here originated with Horn (23) and includes extensions described in Refs. 24–26.

**Image Irradiance Equation.** Image formation is modeled by an image irradiance equation. To standardize the geometry, consider objects to be defined in a left-hand Cartesian coordinate system with the viewing direction aligned with the negative  $Z$  axis. The equation for a visible surface can be given explicitly as  $z = f(x, y)$ . In general, optical systems perform a perspective projection defined with respect to the focal point of the lens. If the size of the objects is small compared to the

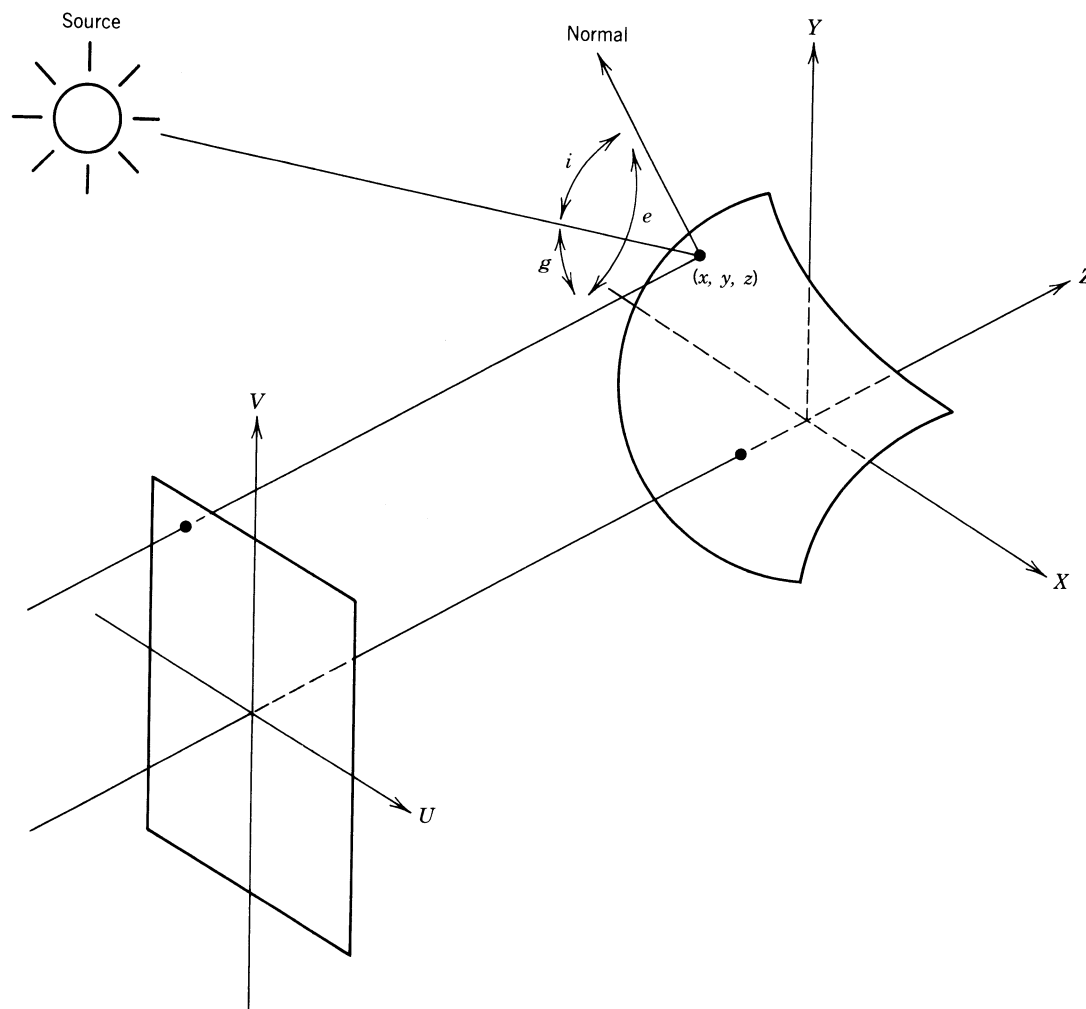
viewing distance, the orthographic projection is obtained, as illustrated in Figure 1. In an orthographic projection all rays from object to image are parallel because the focal point is at infinity. Thus, surface point  $(x, y, z)$  projects to image point  $(u, v)$ . Without loss of generality, let  $u = x$  and  $v = y$ . The orthographic projection, defined here, simply discards the  $z$  coordinate of each visible surface point  $(x, y, z)$ .

There are several ways to specify direction in the coordinate system of Figure 1. Consider points on the unit sphere centered at the origin, called the Gaussian sphere after Hilbert and Cohn-Vossen (27). Each point on the Gaussian sphere identifies the unit vector formed by joining the origin to that point. Thus, standard spherical coordinates can be used to specify the full range of directions. Now, the direction of a viewer facing surface normal at any point on the visible surface  $z = f(x, y)$  can be found by taking the cross product of any two vectors lying in the tangent plane, provided they are not parallel to each other. Two such vectors are  $[1, 0, p]$  and  $[0, 1, q]$  where  $p = \partial f(x, y)/\partial x$  is the slope in  $X$  direction and  $q = \partial f(x, y)/\partial y$  is the slope in the  $Y$  direction. Their cross product is the vector  $[p, q, -1]$ . The quantity  $(p, q)$  is called the gradient of  $z = f(x, y)$  and gradient space is the 2-D space of all such gradients  $(p, q)$ . Gradient space corresponds to the projection of points on the viewer-facing hemisphere of the Gaussian sphere from the origin to the plane  $z = -1$ . Because of this, the gradient space has one drawback. Points where a surface smoothly disappears from view form what Marr (28) called an occluding contour. Points on an occluding contour have surface normals on the Gaussian sphere that intersect the plane  $z = 0$ . These points project to infinity in the gradient space. Another projection of the Gaussian sphere can be used instead. Points on the Gaussian sphere can be projected to the plane  $z = -1$  from the point  $(0, 0, 1)$  rather than from the origin. This is a stereographic projection. Stereographic coordinates will be denoted as  $(f, g)$  to avoid confusion with the gradient  $(p, q)$ . In stereographic coordinates points on an occluding contour lie on the circle  $f^2 + g^2 = 4$ . Equations for transforming between spherical coordinates, the gradient, and stereographic coordinates can be found in Refs. 6 and 25.

Two directions are required to specify the local geometry of the incident and the reflected ray. A total of four parameters are required because each direction has two degrees of freedom. Often, however, one considers materials whose reflectance characteristics are invariant with respect to rotation about the surface normal. For surfaces that are isotropic in this way, only three parameters are required. Figure 1 illustrates one way to define the incident and the reflected ray in terms of three angles  $i$ ,  $e$ , and  $g$ . This choice has one advantage over other possibilities. For a distant viewer and distant light source, the phase angle  $g$  is constant independent of the surface normal.

The amount of light reflected by a surface element depends on its optical properties, its microstructure, and the spectral and spatial distribution of the illumination. The reflectance properties of a surface material are described by its bidirectional reflectance distribution function (BRDF). The BRDF was introduced as a unified notation for the specification of reflectance in terms of the incident and the reflected ray geometry (29). The BRDF identifies an intrinsic property of a surface material. It determines how bright the surface will appear when viewed from a given direction and illuminated from another.

The surface normal relates surface geometry to brightness



**Figure 1.** In an orthographic projection all rays from object surface to image are parallel. With appropriate scaling of the image plane, image coordinates  $(x, y)$  and surface coordinates  $(x, y)$  can be used interchangeably. The incident angle  $i$  is the angle between the incident ray and the surface normal. The emergent angle  $e$  is the angle between the emergent ray and the surface normal. The phase angle  $g$  is the angle between the incident and emergent rays.

because it determines the angles  $i$ ,  $e$ , and  $g$  of Figure 1. For a constant scene irradiance and viewer geometry, the image irradiance measured from a given surface material varies only with the surface orientation. A reflectance map determines image irradiance as a function of surface orientation (23). When the gradient is used to represent surface orientation, the reflectance map is denoted by  $R(p, q)$ . When stereographic coordinates are used, it is denoted by  $R(f, g)$ . A reflectance map is a uniform representation for specifying the reflectance properties of a surface material for a particular light-source distribution and viewer geometry. A reflectance map can be derived analytically for a given BRDF and light-source distribution (30). More commonly, a reflectance map is measured empirically. A calibration object of known shape is used to determine image brightness as a function of surface orientation. The reflectance map obtained this way can be used to analyze other objects made of the same material and viewed under identical conditions of illumination. The empirical approach has the advantage of automatically correcting for the transfer characteristics of the imaging device.

Image formation can thus be described by a single equation called the image irradiance equation. If the gradient is used to

represent surface orientation, the image irradiance equation becomes

$$E(x, y) = R(p, q)$$

where  $E(x, y)$  is the image irradiance at image point  $(x, y)$  and  $R(p, q)$  is the reflectance map value at the corresponding gradient  $(p, q)$ . Shape-from-shading methods reconstruct a surface  $z = f(x, y)$  to satisfy the image irradiance equation.

**Shape from Shading and Occluding Contour.** An object's boundary provides additional information that can be used to establish initial conditions. Parts of the boundary may correspond to sharp edges on the surface, as with polyhedra. Other parts correspond to places where the surface curves around smoothly. The latter defines an occluding contour, as we have seen. At an occluding contour surface orientation is determined locally (12,25,28). In an orthographic projection, a normal to the silhouette in the image plane is also a normal to the surface at the corresponding point on the occluding contour.

Ikeuchi and Horn (25) developed an iterative algorithm to determine surface orientation using the image irradiance equation and a smoothness criterion as constraints. Surface

orientation at occluding contours is the main source of initial conditions although information from singular points, specular points, and self-shadow boundaries can also be included. The stereographic projection is used to represent surface orientation.

Consider the continuous case. The goal is to find functions  $f(x, y)$  and  $g(x, y)$  that make error in the image irradiance equation small while keeping the solution surface as smooth as possible. Departure from smoothness is given by

$$\iint [(f_x^2 + f_y^2) + (g_x^2 + g_y^2)] dx dy$$

where  $f_x, f_y, g_x$ , and  $g_y$  are the first partial derivatives of  $f$  and  $g$  with respect to  $x$  and  $y$ . Error in the image irradiance equation is given by

$$\iint [E(x, y) - R(f, g)]^2 dx dy$$

The problem is to minimize the combined error term  $e$  defined by

$$e = \iint \{ (f_x^2 + f_y^2) + (g_x^2 + g_y^2) + \lambda [E(x, y) - R(f, g)]^2 \} dx dy$$

Error in the image irradiance equation is weighted by  $\lambda$  compared to departure from smoothness. If the reflectance map is known accurately and if the brightness measurements are precise,  $\lambda$  can be made large. On the other hand, if  $\lambda$  is small, a smooth surface is determined despite noise and uncertainties about reflectance and illumination.

Minimization of an integral of the form

$$\iint F(f, g, f_x, f_y, g_x, g_y) dx dy$$

subject to suitable boundary conditions is a problem in the calculus of variations. The function  $F$  is called the performance index. In general, the existence and uniqueness of a solution to problems in the calculus of variations cannot be taken for granted. Careful consideration must also be given to specification of boundary conditions. Correct formulation of a problem requires specifying a performance index that guarantees the existence of a solution and that provides the tightest set of natural-boundary conditions that is consistent with the given data. Much of the early work in shape analysis omitted formal analysis of these problems. Recently, formulations of variational principles, called regularization in the Soviet literature (31), have been developed that guarantee the existence, uniqueness, and stability of the solution. Variational principles are increasingly being applied to problems in computational vision, both retroactively to analyze earlier work and as the basis for new shape-analysis methods (22,32).

In a discrete formulation the local departure from smoothness is given by

$$s_{i,j} = \frac{1}{4}[(f_{i+1,j} - f_{i,j})^2 + (f_{i,j+1} - f_{i,j})^2 + (g_{i+1,j} - g_{i,j})^2 + (g_{i,j+1} - g_{i,j})^2]$$

The local error in the image irradiance is given by

$$r_{i,j} = [E_{i,j} - R(f_{i,j}, g_{i,j})]^2$$

where  $E_{i,j}$  is the measured brightness at image point  $(i, j)$  and  $(f_{i,j}, g_{i,j})$  is the corresponding surface orientation in stereographic coordinates. The problem is to minimize the error term  $e$  given by

$$e = \sum_i \sum_j (s_{i,j} + \lambda r_{i,j})$$

where  $\lambda$  is a free parameter as in the continuous case.

The solution method requires that this last equation be differentiated with respect to  $f_{i,j}$  and  $g_{i,j}$ . For a minimum the partial derivatives are all set to zero, resulting in a large, sparse set of linear equations. These equations are solved using an iterative method, and the set of values for  $f_{i,j}$  and  $g_{i,j}$  determines the solution surface.

Empirical results indicate that the method is both stable and robust. It works well when all information is precise. It continues to work reasonably well even when the reflectance map is only a crude approximation. One problem with this straightforward implementation is that convergence can be very slow. On a fine grid the locations at which initial conditions are specified can be widely separated. The global minimum is achieved by iteratively propagating constraints across the network. A multiresolution algorithm for this formulation of shape from shading has been developed (26) based on multigrid relaxation methods of numerical analysis. Empirical results with the multiresolution implementation suggest that order-of-magnitude gains in efficiency are possible.

**Photometric Stereo.** Another approach to shape from shading uses multiple images to provide additional constraint. These images are taken from the same viewing direction but under different conditions of illumination. This technique is called photometric stereo (33). It allows one to determine surface orientation locally without smoothness assumptions (see also Stereo vision).

Suppose two images  $E_a(x, y)$  and  $E_b(x, y)$  are obtained by varying the direction of illumination. Since there is no change in the imaging geometry, each picture element  $(x, y)$  in the two images corresponds to the same object point and hence to the same gradient  $(p, q)$ . This means that one does not have the problem of first identifying corresponding points in multiple views, as happens in binocular stereo. The effect of varying the direction of illumination is to change the reflectance map  $R(p, q)$  that characterizes the imaging situation.

Let the reflectance maps for the two imaging situations be  $R_a(p, q)$  and  $R_b(p, q)$ , respectively. Suppose a point  $(x_0, y_0)$  has measured intensities  $E_a(x_0, y_0) = \alpha$  and  $E_b(x_0, y_0) = \beta$ . One obtains two equations in the two unknowns  $p$  and  $q$ ,  $R_a(p, q) = \alpha$  and  $R_b(p, q) = \beta$ . There may be more than one solution because the equations are generally nonlinear. Additional information, such as a third image, can be used to determine the correct solution.

The multiple images required for photometric stereo can be obtained by moving a single light source, by using multiple sources individually calibrated, or by rotating the object and imaging device together to simulate the movement of a single light source. An equivalent to photometric stereo can also be achieved in a single view by using multiple sources that can be separated by color.

Photometric stereo is easy to implement. After an initial calibration step, the stereo computation is purely local and may be implemented by table lookup, allowing real-time performance. Photometric stereo is a practical scheme for environments, such as industrial inspection, where the illumination can be controlled. It has been used as the basis for a prototype system to solve the industrial bin-of-parts problem (34).

The gradients computed at neighboring image points may vary considerably due to measurement errors. A smoothness constraint can be used to improve the results of photometric stereo (6). Using a performance index similar to the one dis-

cussed above, the problem is to minimize the error term  $e$  defined by

$$e = \iint \{(f_x^2 + f_y^2) + (g_x^2 + g_y^2) + \sum_i \lambda_i [E_i(x, y) - R_i(f, g)]^2\} dx dy$$

where  $E_i(x, y)$  is the  $i$ th image and  $R_i(f, g)$  is the corresponding reflectance map. The  $\lambda_i$  weigh the errors in the image irradiance equations relative to the departure from smoothness. They may be different if the measurements from the images are not equally reliable.

Photometric stereo is complementary to binocular stereo. Binocular stereo allows the accurate determination of distance to the surface. Photometric stereo determines surface orientation. Binocular stereo works well on rough surfaces with discontinuities in surface orientation. Photometric stereo is best when surfaces are smooth with few discontinuities. Binocular stereo works well on textured surfaces with discontinuities in surface reflectance. Photometric stereo is best when surfaces have uniform optical properties. Photometric stereo has some distinct advantages. There is no difficulty identifying corresponding points in two images because the images are obtained from the same point of view. Determining correspondence is the major computational task of binocular stereo. In certain circumstances surface reflectance can also be found because the effect of surface orientation on image brightness can be removed. Binocular stereo does not provide this capability. In some applications a description of object shape based on surface orientation is preferable to a description based on range.

## Two-Dimensional Shape

In 2-D shape analysis all descriptions are given in terms of image properties alone. It is assumed that there is a direct correspondence between image features and requirements of the task. For some tasks the world is essentially 2-D and free of complications that arise from image projection. Examples include optical character recognition (OCR), inspection of printed circuit boards and VLSI layout, and microscopic blood-cell image analysis. In some robotics tasks the 3-D objects are confined to a small number of stable configurations that can be characterized by the object silhouette. Shape analysis for these tasks typically takes as input a binary image and is based either on object regions or on the object's bounding contour.

**Global Shape Properties of Binary Images.** Some global properties of regions in a binary image can be computed using only local support. Two examples are the metric properties perimeter ( $P$ ) and area ( $A$ ). The ratio  $4\pi A/P^2$  is used as a measure of compactness. The factor of  $4\pi$  in the numerator normalizes the ratio to 1 for a circle, the plane figure that encloses the most area for a given perimeter. The compactness ratio is less than 1 for all other regions. Perimeter, area, and the compactness ratio are invariant under translation and rotation. The compactness ratio is invariant also to changes in scale, provided that the computation of perimeter and area is stable as image resolution changes.

In general, topological properties cannot be computed using only local support. The one exception is the topological property known as Euler number. The Euler number  $E$  of a binary image is the number of connected objects minus the number of

holes. Sometimes it is known a priori that the image contains exactly one object. In this case the Euler number  $E$  can be used to compute the number of holes.

The center of area and the direction of the principal axes are features derived when the position and orientation of an object are needed. A complete theory exists for binary images (6). Special-purpose hardware is commercially available for binary image analysis. This hardware typically combines elementary binary image processing for smoothing, thinning, and noise suppression with the computation of the global features described above. Unfortunately, when objects overlap, the global features computed for the visible portions bear little relation to those that would be computed for the whole object. It is impossible to recognize occluded objects using only global features.

**Other Representations of Two-Dimensional Shape.** Other representations of two-dimensional shape are discussed in Ref. 35. Here, three are briefly described: the Hough transform, the symmetric axis transform, and smoothed local symmetries.

The generalized Hough transform (qv) (36) is one approach to finding instances of occluded shapes. Object shape is expressed parametrically. Local features add their vote to all locations in an accumulator array of parameter values consistent with the evidence provided by the feature. The final tally of votes determines the values of the parameters and hence the shape. Sufficient evidence can be accumulated even if part of the shape is occluded.

The symmetric axis transform (SAT), also called the medial axis transform, can be defined as the locus of centers of maximal disks that touch at least two points on the bounding contour of an object (37). The SAT is information preserving, but it is not stable because it is very sensitive to small perturbations in the bounding contour. In many cases the SAT produces unintuitive descriptions.

Recently, the smoothed local symmetries (SLS) representation has been developed to overcome deficiencies in the SAT in two ways. First, the representation redefines symmetry to be a local property. The precise details are not described here but can be found in Refs. 19 and 38. Potential axes of the shape are the maximal smooth loci of the local symmetries. Second, any axis whose support region is wholly contained in the support region of another axis is deleted. The surviving axes are the smoothed local symmetries and arguably produce a more intuitive description than the SAT. Smoothed local symmetries are an attempt to produce descriptions that can deal with subobjects at a variety of scales rather than at a single level.

## Visible Surfaces

Descriptions at the level of visible surfaces make object properties explicit in the retinocentric coordinate system defined by the viewing direction. Shape properties made explicit include range and surface orientation. Shape from stereo and shape from motion derive range. Shape from shading, shape from contour, and shape from texture derive surface orientation. Discontinuities in range and surface orientation are included if they are available from earlier segmentation processes. Most often, they are derived subsequently. The representation of visible surfaces is included in the intrinsic-image idea of Barrow and Tenenbaum (39) and in the  $2\frac{1}{2}$ -D sketch of Marr (8). Horn refers to a map of surface orientations as a needle diagram (40).

Mackworth (41) popularized the use of gradient space to reason about visible surfaces when interpreting line drawings of polyhedra. More recently Draper (42) analyzed the gradient-space representation for polyhedral scenes. Geometric properties of the gradient space under orthographic and perspective projection are summarized in Ref. (43). Studies of human perception demonstrate that image contours provide cues to 3-D shape that apply in a more general setting (44).

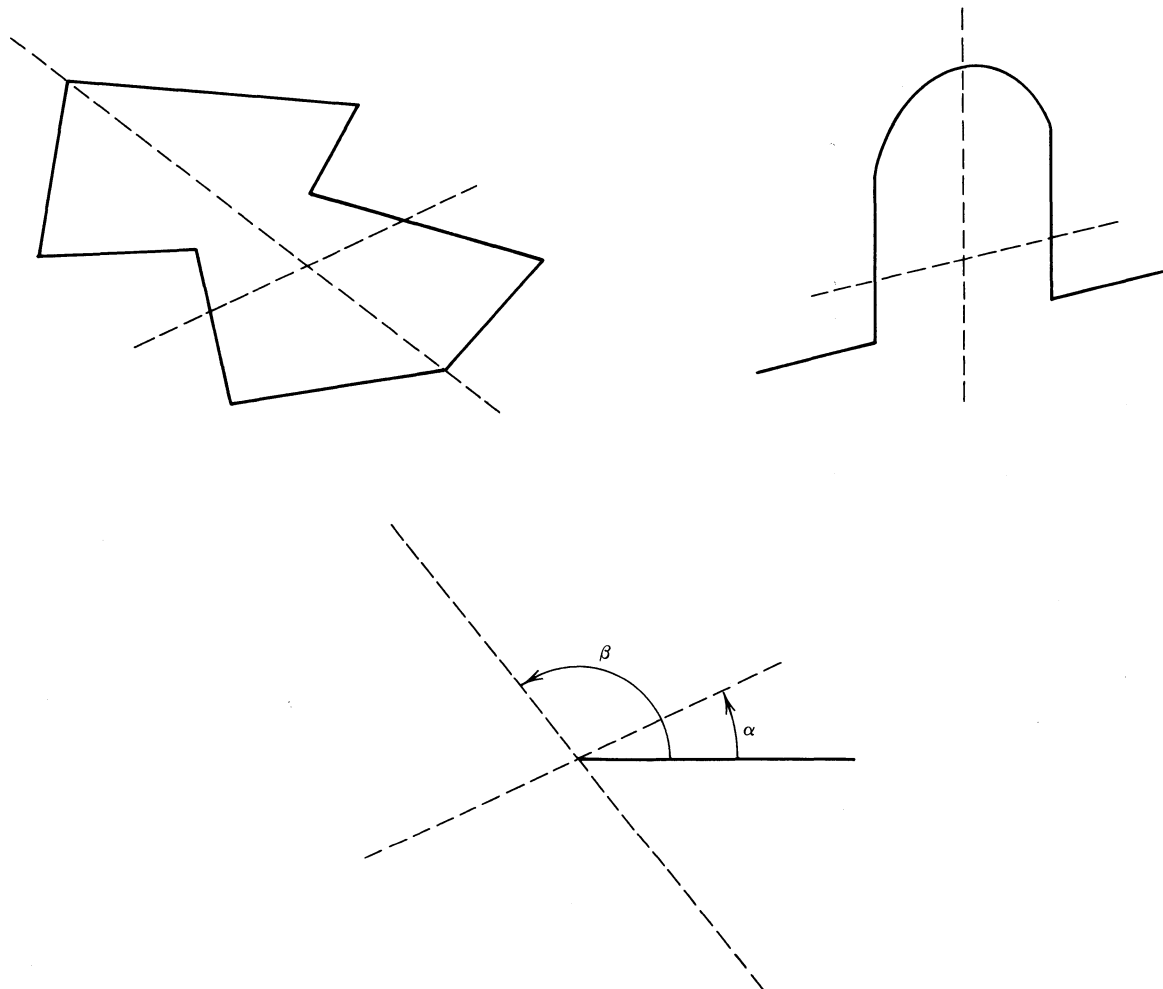
**Skewed Symmetry.** Symmetry in a 2-D shape requires a straight-line axis for which opposite sides are reflective. That is, symmetrical properties occur along lines perpendicular to the axis of symmetry. Kanade (13) defined a generalization called skewed symmetry. Skewed symmetry in a 2-D shape requires symmetrical properties along lines not necessarily perpendicular to the axis, but at a fixed angle to it. Figure 2 illustrates this concept.

Under orthographic projection, a 3-D planar surface with real symmetry appears as a 2-D shape with skewed symmetry. The converse, however, is not always true. Nevertheless, skewed symmetry in an image is often due to real symmetry in the scene. Kanade proposed the following assumption:

*A skewed symmetry depicts a real symmetry viewed from some unknown view angle.*

This assumption is transformed into constraints in the gradient space. It restricts the gradient  $(p, q)$  to lie on a hyperbola in gradient space determined by  $\alpha$  and  $\beta$ . In the absence of global information, Kanade suggested that the minimum-slant interpretation for the gradient is the surface orientation that is perceived. See Ref. 13 for details.

The consequences of Kanade's skew-symmetry assumption can be examined. First, when combined with other constraints, it can lead to a unique global solution for the orientation of each visible surface. The assumption provides a performance index to select a preferred solution from an equivalence class of solutions. The solution determined necessarily describes all skew symmetries as oriented real symmetries. Second, no global solution may be obtained because the equivalence class of solutions may not include one in which all skew symmetries are oriented real symmetries. Third, the solution obtained may not be the physically correct one. Consider, e.g., a scene containing a circular clock mounted on an otherwise blank wall. The projected shape of the clock will have skewed symmetry, depending on the surface orientation  $(p, q)$  of the wall. If one assumes that the clock is circular, local measurements are sufficient to determine the surface orientation of the wall. [There are two solutions of the form  $(p_0, q_0)$  and  $(-p_0, -q_0)$ .] On the other hand, if the clock was elliptical, its projected shape would still have skewed symmetry. This time the



**Figure 2.** Examples of skewed symmetry. A skew symmetry defines two directions: the skewed transverse axis and the skewed symmetry axis. These directions are denoted  $\alpha$  and  $\beta$ .

assumption that the clock was circular would lead to an incorrect determination of the surface orientation of the wall. Fourth, the solution may not agree with human perception. As pointed out in Ref. 14, an ellipse is usually perceived as a tilted circle even though an ellipse has real symmetry.

Brady and Yuille (14) have recently proposed an extremum principle to estimate surface orientation from 2-D contour. The principle selects the plane orientation that maximizes the compactness ratio  $4\pi A/p^2$  introduced earlier. That is, of all true shapes that project to the observed shape, select the one that is the most compact. This is sufficient to determine the gradient  $(p, q)$  of the plane containing the true shape. Kanade's skew-symmetry assumption is implied by this extremum principle. Skew symmetries are interpreted as oriented real symmetries. Only in special cases, however, does the solution correspond to the minimum-slant interpretation.

**Normalized Texture Property Map.** Texture is an important visual cue to the properties of a surface material. At the same time texture gradient is a cue to surface orientation. Kender (18) developed the normalized texture property map (NTPM) to represent the local properties of texture as a function of surface orientation. For convenience, suppose the gradient  $(p, q)$  is used to represent surface orientation, even though, as above, other choices are possible. The result is analogous to a reflectance map  $R(p, q)$ . Local geometric features of the primitive texture elements, called texels, determine equations that the gradient must satisfy. For some textures the problem is locally underconstrained, and methods analogous to those for the shape from shading would be required to determine a solution. Sometimes two or more independent local geometric features can be determined from each texel, each characterized by its own NTPM. Then the situation is analogous to photometric stereo, and surface orientation can be determined locally. There is a strong connection between the local geometric properties of texels and skew symmetry. This connection is explored in Ref. 45.

### Three-Dimensional Shape

Surface-based descriptions can be derived from an image without specific knowledge of the objects in view. However, surface-based descriptions are different from different viewpoints. It is impossible to derive 3-D shape descriptions from an image simply because much of the object is obscured from view. The task in 3-D shape description is to interpret descriptions derived from the image as instances of existing knowledge structures. This is termed model-based vision (3). Model-based vision systems use models to predict what can be seen. This constrains the equivalence class of solutions to include only those solutions expressible within the knowledge base of the system. Some model-based systems, such as ACRONYM and the EGI representation discussed below, model 3-D shapes in a viewpoint-independent way. Many others include specific constraints that are viewpoint-dependent.

There are many representations for solid objects used, e.g., in computer graphics and in computer-aided design and manufacture (CAD/CAM) (46) (see Computer-aided design; Computer-integrated manufacturing). In shape analysis the representation is used to predict how the object will appear. This is necessary to generate an inverted index of possible object features corresponding to a given local image feature (47). With

this additional stipulation the number of 3-D shape representations used in computational vision is much more restricted.

**Generalized Cylinders.** Generalized cylinders were developed by Agin and Binford (48) as a representation of 3-D volumes that emphasizes elongation (see Generalized cylinder representation). A shape is described by sweeping a cross section along an arbitrary 3-D curve, called the spine, and expanding or contracting the cross section according to a scaling function, called the sweeping rule. The cross section need not be circular or at right angles to the spine, although these are simplifications often used in practice. This formulation predates but is similar to the generalized cones of Marr (28). Generalized cones are generalized cylinders whose spines are straight lines.

Generalized cones satisfy many of the criteria for shape representation. There is a natural decomposition into a cross-sectional function and a spine function giving the representation local support. Minor perturbations do make the representation unstable. But in practice, description is preceded by some degree of smoothing to assure stability. The representation is rich and can be applied at multiple scales (20). The representation is object-based and does support a natural segmentation of complex objects into simpler components. It has also been suggested that this representation corresponds to human perception for certain natural structures such as animals (20,47). Unfortunately, not all shapes are represented naturally using generalized cones.

Brooks (49) used generalized cylinders to represent airplane shapes independent of viewpoint in a system called ACRONYM. ACRONYM uses part-whole graphs of generalized cylinders to predict the appearance of wide-bodied aircraft in aerial photographs. Projections of generalized cylinders are called ribbons. ACRONYM determines the viewpoint simultaneously with the interpretation of ribbons as projected generalized cylinders.

**Extended Gaussian Image.** The extended Gaussian image (EGI) of an object records the variation of surface area with surface orientation (50). The EGI is invariant to translation and can be normalized to be invariant also to scale. The EGI is typically represented as a function defined on the Gaussian sphere. Rotations are easy to deal with since an object and its EGI rotate together. The EGI is easy to derive from other representations of 3-D objects.

The EGI uniquely represents convex objects and is thus information preserving for this class of objects. An iterative algorithm has been developed to reconstruct a convex polyhedron from its EGI (51). The EGI has also been applied to determine object attitude, i.e., the 3-D rotation required to bring a sample object into correspondence with a prototype. The visible hemisphere of a convex object's EGI can be computed from the visible-surface description of surface orientation. Conceptually, determining object identity corresponds to finding a prototype EGI with an identical hemisphere. Determining object attitude corresponds to matching the position and orientation of the visible hemisphere to the prototype EGI. This approach has been applied to the industrial problem of picking parts out of a bin (34). The matching computation can be ill-conditioned. Recently, the mixed-volume function (introduced in Ref. 51) has been used as a similarity measure for attitude determination. The result is more stable than direct EGI



matching and can support efficient multiresolution attitude determination (52).

## Conclusions

At first glance, shape analysis in computational vision can appear to be a collection of ad-hoc techniques. In many applications the environment can be controlled and the computational task can be structured so that a special-purpose vision system will succeed. Special-purpose systems represent an extremely limited repertoire of objects so that simple descriptions often are sufficient to distinguish between them. In contrast, as Binford (53) says, "general vision requires strong description and weak classification." This entry presents a coherent framework for shape analysis in a general-purpose vision system based on a thorough understanding of image formation, a characterization of the computational task, criteria for representations of shape, and correspondence with human perception. These are the themes. But much remains before they will be achieved in practice.

Shape from shading was used as the major illustrative example. The methods described satisfy many of the criteria for representation of shape. Surface orientation is computable using local support. The methods based on variational formulations are stable. The representation is rich because the orientation map produced is dense. Description at multiple scales has not explicitly been addressed. But the choice of  $\lambda$  indirectly influences the level of detail considered. The generation of representations at multiple scales is an obvious application of the multigrid implementation approach. The final description is in terms of an intrinsic object property. The visible surface orientation map can be extended to a full 3-D representation through the EGI, although this is currently limited to simple objects. There is no claim that the shape-from-shading methods correspond to human performance.

The prerequisite problem of how to segment an image to obtain a contour description has not been considered. This problem is generally considered separately as edge detection (qv) or region growing (see Region-based segmentation). Some argue that segmentation and shape analysis cannot be separated but are inherently part of the same problem. See, e.g., Refs. 53 and 54. Segmentation issues arise again at a higher level. Initial representations for shape are typically dense. It is desirable to segment shape descriptions both to be more storage efficient and to provide symbolic descriptions to facilitate subsequent analysis. There are a myriad of segmentation schemes proposed at each level of shape representation: 2-D shape, visible surfaces, and 3-D shape. Most address storage efficiency foremost, but others deal with aspects related to symbolic description. This is an area of ongoing research.

## BIBLIOGRAPHY

1. H. G. Barrow and J. M. Tenenbaum, "Computational vision," *Proc. IEEE* **69**, 572-595 (1981).
2. M. Brady, "Computational approaches to image understanding," *ACM Comput. Surv.* **14**, 3-72 (1982).
3. T. O. Binford, "Survey of model-based image analysis systems," *Int. J. Robot. Res.* **1**(1), 18-64 (1982).
4. D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
5. R. Nevatia, *Machine Perception*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
6. B. K. P. Horn, *Robot Vision*, MIT Press, Cambridge, MA, 1986.
7. M. Brady, "Artificial intelligence and robotics," *Artif. Intell.* **26**, 79-121 (1985).
8. D. Marr, *Vision*, Freeman, San Francisco, CA, 1982.
9. H. H. Baker and T. O. Binford, Depth from Edge and Intensity Based Stereo, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, BC, pp. 631-636, 1981.
10. W. E. L. Grimson, *From Images to Surfaces*, MIT Press, Cambridge, MA, 1981.
11. A. P. Witkin, "Recovering surface shape and orientation from texture," *Artif. Intell.* **17**, 17-45 (1981).
12. H. G. Barrow and J. M. Tenenbaum, "Interpreting line drawings as three-dimensional surfaces," *Artif. Intell.* **17**, 75-116 (1981).
13. T. Kanade, "Recovery of the three-dimensional shape of an object from a single view," *Artif. Intell.* **17**, 409-460 (1981).
14. M. Brady and A. Yuille, "An extremum principle for shape from contour," *IEEE Trans. Patt. Anal. Mach. Intell.* **6**, 288-301 (1984).
15. T. O. Binford, "Inferring surfaces from images," *Artif. Intell.* **17**, 205-244 (1981).
16. E. C. Hildreth, "Computations underlying the measurement of visual motion," *Artif. Intell.* **23**, 309-354 (1984).
17. B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.* **17**, 185-203 (1981).
18. J. R. Kender, A Computational Paradigm for Deriving Local Surface Orientation from Local Texture Properties, *Proceedings of the IEEE Workshop on Computer Vision: Representation and Control*, Rindge, NH, pp. 143-152, 1982.
19. M. Brady, Criteria for Representations of Shape, in J. Beck, B. Hope, and A. Rosenfeld (eds.), *Human and Machine Vision*, Academic Press, New York, 1983.
20. D. Marr and H. K. Nishihara, "Representation and recognition of the spatial organization of three dimensional structure," *Proc. Roy. Soc. Lond. B* **200**, 269-294 (1978).
21. F. Mokhtarian and A. K. Mackworth, "Scale-based description and recognition of planar curves and two-dimensional shapes," *IEEE Trans. Patt. Anal. Mach. Intell.* **8**, 34-43 (1986).
22. M. Brady and B. K. P. Horn, "Rotationally symmetric operators for surface interpolation," *Comput. Vis. Graph. Im. Proc.* **22**, 70-94 (1983).
23. B. K. P. Horn, "Understanding image intensities," *Artif. Intell.* **8**, 201-231 (1977).
24. R. J. Woodham, "Analysing images of curved surfaces," *Artif. Intell.* **17**, 117-140 (1981).
25. K. Ikeuchi and B. K. P. Horn, "Numerical shape from shading and occluding boundaries," *Artif. Intell.* **17**, 141-184 (1981).
26. D. Terzopoulos, "Image analysis using multigrid relaxation methods," *IEEE Trans. Patt. Anal. Mach. Intell.* **8**, 129-139 (1986).
27. D. Hilbert and S. Cohn-Vossen, *Geometry and the Imagination*, Chelsea, New York, 1952.
28. D. Marr, "Analysis of occluding contour," *Proc. Roy. Soc. Lond. B* **197**, 441-475 (1977).
29. F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis, Geometrical Considerations and Nomenclature for Reflectance, *NBS Monograph* 160, National Bureau of Standards, Washington, DC, 1977.
30. B. K. P. Horn and R. W. Sjoberg, "Calculating the reflectance map," *Appl. Opt.* **18**, 1770-1779 (1979).
31. A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, Winston, New York, 1977.

32. T. Poggio and V. Torre, Ill-Posed Problems and Regularization Analysis in Early Vision, AI Memo 773, MIT AI Lab, Cambridge, MA, 1984.
33. R. J. Woodham, "Photometric method for determining surface orientation from multiple images," *Opt. Eng.* **19**, 139–144 (1980).
34. B. K. P. Horn and K. Ikeuchi, "The mechanical manipulation of randomly oriented parts," *Sci. Am.* **251**, 100–111 (1984).
35. Reference 4, Chapter 8.
36. D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Patt. Recog.* **13**, 111–12 (1981).
37. H. Blum and R. N. Nagel, "Shape description using weighted symmetric axis transform," *Patt. Recog.* **10**, 167–180 (1978).
38. M. Brady and H. Asada, Smoothed Local Symmetries and their Implementation, in M. Brady and R. Paul (eds.), *Robotics Research: The First International Symposium*, MIT Press, Cambridge, MA, pp. 331–354, 1984.
39. H. G. Barrow and J. M. Tenenbaum, Recovering Intrinsic Scene Characteristics from Images, in A. R. Hanson and E. M. Riseman (eds.), *Computer Vision Systems*, Academic Press, New York, pp. 3–26, 1978.
40. B. K. P. Horn, Sequins and Quills—Representations for Surface Topography, AI Memo 536, MIT AI Lab, Cambridge, MA, 1979.
41. A. K. Mackworth, "Interpreting pictures of polyhedral scenes," *Artif. Intell.* **4**, 121–137 (1973).
42. S. W. Draper, "The use of gradient and dual space in line-drawing interpretation," *Artif. Intell.* **17**, 461–508 (1981).
43. S. A. Shafer, T. Kanade, and J. R. Kender, "Gradient space under orthography and perspective," *Comput. Vis. Graph. Im. Proc.* **24**, 182–199 (1983).
44. K. A. Stevens, "The visual interpretation of surface contours," *Artif. Intell.* **17**, 47–75 (1981).
45. T. Kanade and J. R. Kender, Mapping Image Properties into Shape Constraints: Skewed Symmetry, Affine-Transformable Patterns, and the Shape-from-Texture Paradigm, in J. Beck, B. Hope, and A. Rosenfeld (eds.), *Human and Machine Vision*, Academic Press, New York, pp. 237–257, 1983.
46. A. A. G. Requicha, "Representations for rigid solids: Theory, methods, and systems," *ACM Comput. Surv.* **12**, 437–464 (1980).
47. R. Nevatia and T. O. Binford, "Description and recognition of curved objects," *Artif. Intell.* **8**, 77–88 (1977).
48. G. J. Agin and T. O. Binford, Computer Description of Curved Objects, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Palo Alto, CA, pp. 629–640, 1973.
49. R. A. Brooks, "Symbolic reasoning among 3-D models and 2-D images," *Artif. Intell.* **17**, 285–348 (1981).
50. B. K. P. Horn, "Extended Gaussian images," *Proc. IEEE* **72**, 1671–1686 (1984).
51. J. J. Little, An Iterative Method for Reconstructing Convex Polyhedra from Extended Gaussian Images, *Proceedings of the Third National Conference on Artificial Intelligence*, Washington, DC, pp. 247–250, 1983.
52. J. J. Little, Determining Object Attitude from Extended Gaussian Images, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 960–963, 1985.
53. Reference 3, p. 60.
54. J. M. Tenenbaum and H. G. Barrow, "Experiments in interpretation-guided segmentation," *Artif. Intell.* **8**, 241–274 (1977).

R. J. WOODHAM  
University of British Columbia

SHAPE FROM SHADING. See Shape analysis; Vision, early.

## SHRDLU

SHRDLU is a natural-language-understanding (qv) program that was developed by Winograd for his dissertation (T. Winograd, *Understanding Natural Language*, Academic Press, New York, 1972). SHRDLU understands sentences limited to the blocks-world domain. Knowledge is represented in procedural form using two special extensions of LISP, PROGRAMMAR, and PLANNER (qv). PROGRAMMAR was written by Winograd to deal with syntactic parsing (qv). Meanings are represented in Hewitt's PLANNER (C. Hewitt, PLANNER: A Language for Proving Theorems in Robots, *Proceedings of the First International Joint Conference on Artificial Intelligence*, Washington, DC, pp. 295–301, 1969). Winograd's treatment of syntax is based on Halliday's systemic grammar [see M. A. K. Halliday, "Categories of the theory of grammar," *Word* **17**, 241–292 (1961); and Language Structure and Language Function, in J. Lyons (ed.), *New Horizons in Linguistics*, Penguin Books, Harmondsworth, UK, pp. 140–165, 1970]. SHRDLU reached at its time an unprecedented level of performance in language understanding.

J. GELLER  
SUNY at Buffalo

## SIMULA

SIMULA (SIMULATION LAnguage) was developed as an extension of ALGOL-60. It permits the definition of processes that can be executed quasi-parallel. Every process has its own data set and a set of instructions. SIMULA introduced the concept of "class" into programming languages. Single processes are instances of a possibly user-defined class. SIMULA had a big impact on many modern programming languages, e.g., SMALLTALK (qv) and ADA. An introduction to the process aspects of SIMULA is given by O-J. Dahl and K. Nygaard, "SIMULA an ALGOL-based simulation language," *CACM* **9**, 671–678 (September, 1966). Information about SIMULA67 can be found in O-J. Dahl, B. Myhrhaug, and K. Nygaard, *The SIMULA67 Common Base Language*, Publication no. S-2, Norwegian Computing Center, Forskningsveien 1B, Oslo, 1968.

J. GELLER  
SUNY at Buffalo

## SIR

A prototype natural-language-understanding (qv) system driven by a LISP-written sentence pattern matcher, SIR was written by Raphael at MIT [see B. Raphael, SIR: A Computer Program for Semantic Information Retrieval, in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, pp. 33–145, 1968].

A. HANYONG YUHAN  
SUNY at Buffalo

## SLIP

A list-processing utility implemented as a FORTRAN subroutine package written by Weizenbaum in 1963, SLIP character-

istically uses doubly linked lists as basic data objects and is embedded in the overt context of its host FORTRAN [see J. Weizenbaum, "Symmetric list processing," *CACM* 6, 524-544 (1963), and N. Findler, J. Pfaltz, and H. Bernstein, *Four High Level Extensions of FORTRAN IV: SLIP, AMPPL-II, TREE-TRAN, and SYMBOLANG*, Spartan, New York, 1971].

A. HANYONG YUHAN  
SUNY at Buffalo

## SMALLTALK

SMALLTALK is a programming language that introduced the concept of "object-oriented programming" (qv) and is based on the programming languages SIMULA (qv) and to a lesser extent LISP (qv). SMALLTALK was designed originally for a personal computer with the goal to have a single interface to operating system, language, and graphics. A small number of concepts is used universally: Objects are instances of "classes" that are themselves objects, and objects can inherit code ("methods") from their classes. Instead of procedure calls, "message passing" is used as control structure. In this way the user specifies what he wants rather than how to do it. SMALLTALK-80 is described by A. Goldberg and D. Robson, *Smalltalk-80: the Language and its Implementation*, Addison-Wesley, Reading, MA, 1983. A description of the SMALLTALK user interface can be found in L. Tesler, "The Smalltalk environment," *Byte* 6, 90-147 (August, 1981).

J. GELLER  
SUNY at Buffalo

## SNePS

SNePS, the Semantic Network Processing System, was developed by Shapiro (see S. C. Shapiro, The SNePS Semantic Network Processing System, in N. V. Findler (ed.), *Associative Networks: The Representation and Use of Knowledge by Computers*, Academic Press, New York, pp. 179-203, 1979); S. C. Shapiro and W. J. Rapaport, "SNePS Considered as a Fully Intensional Propositional Semantic Network," *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA, pp. 278-283, (1986). SNePS is a fully intensional system of knowledge representation [see A. S. Maida and S. C. Shapiro, "Intensional concepts in propositional semantic networks," *Cog. Sci.* 6, 291-330 (1982)]. It incorporates bidirectional inference (see S. C. Shapiro, J. Martins, and D. P. McKay, Bi-directional Inference, *Proceedings of the Fourth Annual Conference of the Cognitive Science Society*, Program in cognitive science, University of Michigan, Ann Arbor, MI, pp. 90-93, 1982) and reasoning with recursive rules (see D. P. McKay and S. C. Shapiro, Using Active Connection Graphs for Reasoning with Recursive Rules, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., pp. 368-374, 1981). SNePS permits path-based and node-based inference (qv) and contains a full implementation of predicate logic, including nonstandard connectives and quantifiers. Several interfaces have been built, permitting a limited natural-language and a logic-programming-oriented communication with the system (see D. P. McKay and J. Martins, SNePSLOG User's Manual, SNeRG Technical Note. 4,

State University of New York at Buffalo, Department of Computer Science, 1981).

J. GELLER  
SUNY at Buffalo

## SNIFFER

A network-deduction system developed around 1977 by Fikes and Hendrix at SRI International, SNIFFER has embedded heuristics (qv) that provide guidance about the order and method for matching network elements. SNIFFER has the general power of a theorem prover (see Theorem proving) for making deductions on the network database (see R. Fikes and G. Hendrix, A Network-Based Knowledge Representation and Its Natural Deduction System, *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, MA, pp. 235-246, 1977).

K. S. ARORA  
SUNY at Buffalo

## SNOBOL-4

SNOBOL-4 is a string-and-pattern-oriented programming language. It is described in R. E. Griswold, J. F. Poage, and I. P. Polonsky, *The SNOBOL-4 Programming Language*, Prentice-Hall, Englewood Cliffs, NJ, 1971. The superior qualities of SNOBOL-4 and its predecessors as a pattern-matching (qv) language have been used to implement several AI programs (e.g., see S. C. Shapiro and G. H. Woodmansee, A Net Structure Based Relational Question Answerer: Description and Examples, in *Proceedings of the First International Joint Conference on Artificial Intelligence*, Washington, DC, pp. 325-346, 1969). SNOBOL-4 is very different from most other programming-language families. Variables are not typed, there is no block structure, and programs are interpreted, but it is an imperative language. It permits programmer-defined data types and embodies "tables" that are arrays that can be indexed by elements of any data type, e.g., by strings. Elaborate tracing mechanisms are supplied by the language.

J. GELLER  
SUNY at Buffalo

## SOCIAL ISSUES OF AI

Even amidst the hyperbole that surrounds the subject of computers, AI is the vanguard of computer fields in both apparent possibilities for significant social change (positive and negative) and in claims and expectations about such changes. Since its founding, both AI researchers and the media have speculated about the social implications of AI in terms ranging from animated to sensational, and major recent announcements such as the Japanese Fifth Generation Computing Program (see Computer systems; Logic programming) have added to the intensity of interest in AI and society.

As some have noted, the term "artificial intelligence" was itself a rather controversial assertion about the early expectations for the field. Although some in the AI community have considerably toned down their early enthusiasm, others have

continued to make profound social claims for AI. Feigenbaum and McCorduck (1), e.g., said in their 1983 book that with powerful AI-based computers "revolution, transformation, and salvation are all to be carried out." And the popular press frequently echoes this kind of rhetoric. A 1980 *New York Times Magazine* article (2), e.g., promises "Computers able to replace doctors, lawyers, stockbrokers, journalists are some of the implications of a future with artificial intelligence."

So far debate on the social implications of AI has consisted primarily of speculation and somewhat ideological arguments about the role of technology in society. That is, there are on the one hand those who believe that AI in the range of human intelligence is inevitable and that society had better adjust to it; on the other hand are those who argue that AI is either not possible in the most sophisticated sense or it ought not to be pursued. There are some who hold intermediate positions, though they are considerably less visible. Although it is useful to be aware of these broad claims and general positions, it is also possible to gain insight about social issues of AI from both historical comparisons and serious empirical and sociological work on the effects of AI, which is just beginning.

This entry provides a working notion of the boundaries of AI for this analysis (see also Limits of AI) and categorizes the impacts to be discussed; gives a brief outline of how AI theory and applications might develop in order to assess the nature of potential social impacts; outlines general schools of thought on the social impacts of AI and assesses their relation to general notions of technology's relation to society; and analyzes specific potential economic, social control, and psychological impacts of AI use.

### Categorizing AI and Its Impacts

This evaluation of social implications is based on a broad definition of AI, since AI is not really a distinct field of research as much as a family of related areas that share common goals and research techniques. These goals are a desire to understand human intelligence through the building of computer models and/or a desire to build machines that can tackle complex indeterminate problems, no matter whether the machines elucidate human intelligence. The techniques used include the use of symbolic (as opposed to numeric) models and programming languages (usually LISP and PROLOG), a focus on expanding the power of computer programs through the logic and structure of these symbolic models, and a reliance on an interdisciplinary set of ideas and theories from philosophy, psychology, mathematics, computer science, and in some cases neuroscience. Figure 1, from Fleck (3), is a helpful map of the subfields of AI and some of their connections to other disciplines.

AI means different things to different people. To a businessman it may be synonymous with expert systems (qv); to some AI researchers only the most basic work—e.g., theoretical explorations of knowledge representation (qv)—is included in their view of the field. Thus, the social issues associated with AI will vary depending on one's conception of the boundaries (and the core) of the field. Taking the reasonably broad view of AI outlined in the paragraph above, the key social issues can be put into the following categories.

**Economic Issues.** How many jobs will be displaced by AI, particularly robotics (qv) and expert systems? What will the remaining jobs entail? Will most people continue to be em-

ployed in conventional jobs, or will AI eventually result in a much lower proportion of the population at work or a shorter work week? If the economy can produce useful goods and services with relatively few workers, who will buy those goods and services? How will the transition to such an economy occur?

**Political and Control Issues.** Will intelligent computer systems shift the control of important social decisions—e.g., military decisions (see Military, applications in), business decisions, decisions about eligibility for social programs, judicial decisions (see Law applications)—to machines? How will it affect the nature and quality of those decisions? In what ways could or should people ultimately control the decisions made by such machines?

**Psychological and Sociological Issues.** If machines can approach human intelligence levels in significant ways, how will that affect people's conception of themselves? If AI is used in such a way that people are essentially subordinate to machines, what will the psychological impacts be? As advanced computers encroach on more and more aspects of everyday life, how will lifestyles—e.g., patterns of interaction, kinds of activities, balance of work and leisure—change?

**Technical Spin-off Issues.** How will AI enhance or change other sciences and technologies? Will AI-based computational models improve (or muddle) psychological theory and our understanding of the mind? To what extent will AI applications enhance, e.g., medicine (see Medical advice systems), chemistry (see Chemistry, AI in), or education (see Education applications)?

### Range of Possibilities for AI's Future Directions

Clearly, it is useful to have some sense of how AI might develop and who might use AI in order to judge the nature and order of magnitude of potential impacts. Just as clearly, describing the future of a technology is a risky and problematic endeavor. Some caveats should be kept in mind:

1. breakthroughs are unpredictable in nature and in timing;
2. the nontechnical factors that affect the development and use of a technology are complex;
3. AI theory and technology are both in their infancy; and
4. previous predictions for AI have been notoriously inaccurate.

With these uncertainties in mind, Wieckert and Dray (4) undertook an analysis of AI's potential development that relied in part on interviews with approximately two dozen key AI researchers. The study assessed possible futures for AI at a time in the early first half of the next century when AI technology would be "relatively advanced and mature." Despite the difficulties of such a projection, the study found substantial agreement about the principal problems to be addressed and the range of possible futures. Although the study can only be considered suggestive, the following results are instructive.

The vast majority of AI researchers seem to believe in what has been called the "strong form" of Church's thesis (qv). Basically, the assertion is that any effective procedure or set of steps can be programmed on a computer and that everything humans do can be expressed as effective procedures. This im-

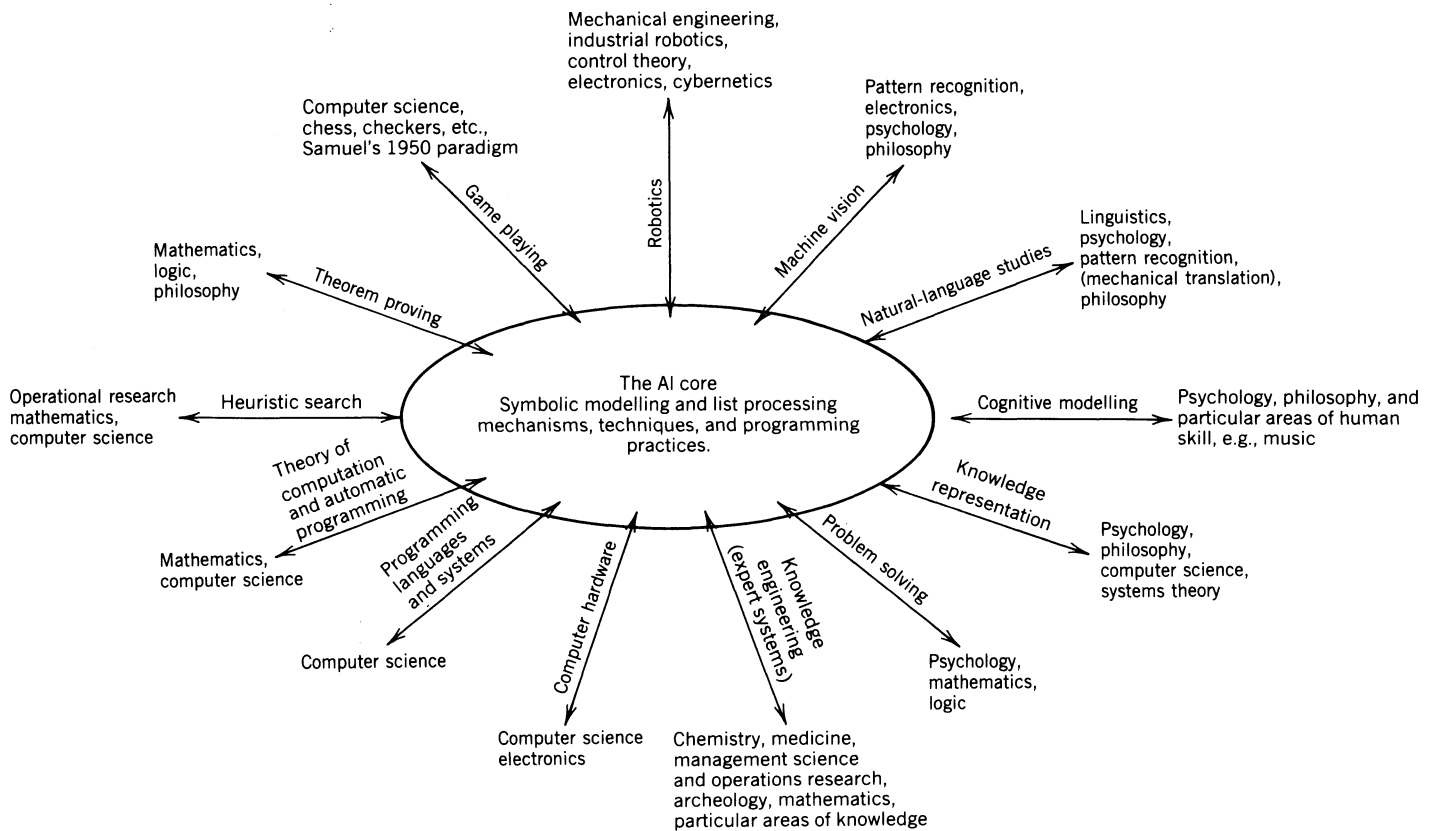


Figure 1. Schematic representation of the subfields of AI. From Ref. 3.

plies that there are few fundamental barriers for machine intelligence and thus that over the long term intelligent machines would meet or surpass humans in most cognitive skills. There was a wide range of sophistication and style in the way this view was expressed. For some, AI research would be less enjoyable if they admitted fundamental barriers. At the same time many added caveats to this belief. A few said they had a gut feeling that there are some aspects of humanity that cannot be reproduced in a digital computer, e.g., "randomness," "intuition," or "soul." Others argued that they could see no purpose in instilling into computers certain aspects of humanity, e.g., love or sadness. Still others noted that human intelligence may be so complex as to frustrate any foreseeable attempts to model that intelligence on a computer. One senior researcher noted, "People are mechanistic, but the mechanisms may be subtle. There may be such subtlety that the fact that they're mechanistic is irrelevant for modeling."

The key uncertainty AI researchers expressed about long-range futures was the breadth of applicability for AI systems. That is, each researcher was near certain that some systems would exceed human cognitive ability in specific areas, e.g., medical diagnosis or specific vision (qv) applications. The question then arises: For how broad an application area could a given system be useful, or how broad a knowledge base can be constructed in a single system? If AI systems continue with little more breadth than the current systems have (e.g., interpreting NMR spectra or oil dipmeter data), they would be useful and perhaps even cause significant social change but would fall far short of intelligence in more than a very narrow sense. If, on the other hand, a system can be designed that is competent in the entire field of medicine (perhaps emphasizing a particular specialty) or that interprets an unconstrained set of

visual signals or natural-language utterances, the implications would be far greater.

The study found wide agreement that commonsense reasoning (qv), knowledge representation (qv), and learning (qv) are the problems at the core of fundamental long-range progress in AI. Some ordered the importance of these problems differently, whereas others argued that one of the three is the "key," and the other problems are mere subsets.

A second set of key problems are more focused on application areas for AI. They are closely related to and draw heavily from research in the above broader areas, but to some extent they are also independent problem domains. These include language processing; real-time, 3-D vision; robust algorithms and structures for robot control and mobility; and broadly applicable, "expert-system" assistants.

Table 1 briefly describes the range of possible futures as identified by the study for development of AI in three of the key application areas: natural language, expert/reasoning systems, and vision/robotics. Three possible levels of progress are described for each area: One assumes a "minimum" amount of progress will be made in fundamental problems of AI (learning, knowledge representation, and commonsense reasoning); a second assumes "moderate" progress; and a third assumes "maximum" progress. Factors that could influence which of the three scenarios is more accurate include the level of future research funding, the balance between basic and applied research, and the unpredictable appearance of what researchers called an "AI Newton," who would make a series of fundamental advances and guide AI as a discipline. (Though it may be tempting to imagine such a savior, it is questionable to what extent one brilliant individual really changes the course of science; history and sociology of science tends to show that

Table 1. Scenarios for Progress in Three AI Application Areas<sup>a</sup>

<i>Natural Language</i>	
<i>Minimum.</i> Systems will be available to interpret continuous speech for somewhat restricted subject areas and with good acoustics (e.g., a microphone instead of a telephone). Real-time spoken-language translation (e.g., from a telephone in one country to a telephone in the other) will be available in a very primitive form. Highly specialized written text could be categorized and placed in databases.	<i>Moderate.</i> Expert systems will still be somewhat restricted in subject area but extremely powerful and in some cases will be given de facto control over routine decisions. For instance, an expert system (or network of expert systems) might essentially have mastery over most of the field of medicine. It would learn by reading articles. It would not deal with ethical issues. Expert systems would substantially augment individuals' knowledge and access to knowledge.
<i>Moderate.</i> Systems will be able to interpret continuous speech and unrestricted written language in defined domains with only a few frustrations and problems caused by gaps in the database. These systems will be supported by massive databases enabled by cheap memory and advances in knowledge-representation techniques. These databases will also be created through these natural-language systems.	<i>Maximum.</i> Intelligent machines would be a near-constant companion and assistant to most humans, essentially resulting in a new level of intelligence through human-machine synthesis. The machine would know its associated individual well and would help satisfy some of their problems, emotions, need for conversations, etc. Intelligent machines might even have a direct connection to the brain and would augment human intelligence enormously. According to the individual's choice, the machine could perform certain tasks more or less autonomously, such as paying bills, maintaining the house, etc.
<i>Maximum.</i> The fundamental problems of representation of knowledge, commonsense reasoning, and learning would essentially be solved, perhaps as a result of the appearance of one or more "AI Newtons." The result would be a computer system whose language proficiency would be better than many humans, perhaps like that of Hal 9000 in 2001: <i>A Space Odyssey</i> . (One researcher characterized the difficulty of this problem as "like saying we could solve the teleportation problem in 50 years. It just requires figuring out what molecules are in an object and duplicating those molecules somewhere else.") It would be possible to assemble enough computing power to match a human brain, although intelligent machines would not necessarily be like humans.	
<i>Expert Systems</i>	
<i>Minimum.</i> Expert systems will be widely deployed as decision aids. They will be extremely limited in subject area. They will have a natural-language interface that will process inquiries related to their domain with some constraints on vocabulary and syntax.	<i>Robotics</i>
	<i>Minimum.</i> Some robots will be mobile, but we will not depend heavily on mobile robots. Virtually all manufacturing tasks could be automated. The problems of mobility and control algorithms for robotic devices will not be fully solved, but robots will be used widely in warfare. Similarly, vision will improve but still will be somewhat primitive compared to humans.
	<i>Moderate.</i> Mobile robots will be prevalent, both in warfare and in the service industries as waitresses, toll-takers, drivers. Household robots will be common and cheap and will perform easy tasks around the house.
	<i>Maximum.</i> Robots would be freely mobile with on-board computers, and vision would be in some cases more accurate and powerful than that of humans.

<sup>a</sup> Reference 4. Note: The target date for these scenarios was specified as a time in the early first half of the next century, when AI technology would be relatively advanced and mature.

science advances primarily through the work of groups of people.)

Although it should be emphasized that this study is only a brief and tentative sketch of alternative possibilities for AI, the scenarios depicted provide an interesting starting point for an analysis of social impacts. Two points are especially relevant to this analysis:

1. One that may seem obvious but is worth emphasizing is that expectations for AI's social impacts vary widely according to one's assumptions about the rate of progress on fundamental theoretical problems. This is not a linear relationship, however; technologies based only on current theory could have significant social ramifications if they are pervasive.
2. Although the "maximum" scenario—with fluent computers, customized expert-system companions, and robots with human-level vision and mobility—is most often depicted as likely in the popular press, it depends on tremendous advances in theoretical understanding that are impossible to predict and may not happen at all.

#### General Schools of Thought on AI's Social Impacts

As noted above, the serious literature and thinking on AI's social impacts is rather skimpy. In fact, until perhaps the

early 1980s one could say with only minor simplification that there were three traditional camps.

**The AI Believers.** The first traditional camp consists largely of AI practitioners who are strongly optimistic about the results of AI research and see profound (mostly positive, but in any case inevitable) social changes as a result. These changes include relieving humans from nearly all routine or menial tasks and perhaps many of the more complicated tasks as well; helping make more rational decisions in government or the judicial system; improving and customizing education; and helping to empower citizens and democratize society through more powerful computers accessible to the layman. Some of the most distinguished members of the AI community are (or have been) in this camp, including Herbert Simon and Allen Newell (5), Marvin Minsky (6), Edward Feigenbaum (1), and others. There have been few in-depth studies that present these arguments; in many cases they are presented in the prefaces of books or the postscripts of research papers or in short articles for panels or the popular press (7). Many of these views are represented in McCorduck's history of AI (8). Nilsson (9) has recently written a more detailed work illustrative of this category. In it he argues that AI is likely to result in considerably less work for people and that society should welcome that outcome instead of fearing it and develop new leisure activities and new systems for distributing wealth in an



automated society. Nilsson's paper is discussed in more detail below under Economic Issues.

**The Social Critics.** Another traditional camp tends to hold the view that although AI may eventually succeed technically at much of what it attempts, many of the social implications will either be negative or will simply reinforce existing power structures in society. Weizenbaum (10) is the most eloquent proponent of this view. He argues that Western society has become enamored with mechanistic models, neglecting its own choices and its humanity; that many of the proposed applications of AI deny human choice and ignore moral issues; and that certain research topics ought not to be pursued. He describes such topics as those that are "obscene" and "represent an attack on life itself" (he gives the example of a proposal for coupling an animal's visual system and a brain to a computer); those that "propose to substitute a computer system for a human function that involves interpersonal respect, understanding and love" (he gives the example of a computerized psychotherapist); and those that "can easily be seen to have irreversible and not entirely foreseeable side effects," particularly if they are not tied to a pressing human need.

**The Disbelievers.** A third group argues that many of the goals of AI are not possible at all, or not possible in the way they are framed by researchers. H. L. Dreyfus (11) is the most visible of these writers. He argues that "the boundary [for progress in AI] is near" and that "the results [up to now] have been sufficiently disappointing to be incriminating." Dreyfus argues that fundamental progress in AI has been stymied by what he considers four erroneous assumptions in their approach to a thinking machine: that a digital computer resembles the brain in the way it handles information; that the brain processes information as a computer does, at some level; that human knowledge and behavior is formalizable; and that knowledge can be meaningful in discrete chunks. In a recent expansion of Dreyfus's earlier work he and his brother, S. Dreyfus, argue that expert systems could reach competence in certain narrow domains but could not reach "proficiency" or "expertise," which requires common sense and possibly holistic mental processes (12). Searle (13) could also be considered a disbeliever. Arguing that a computer program alone could not result in genuinely intelligent behavior, Searle compared the program to a man locked in a room who answers Chinese questions by following rules (or a computer program) for responding to Chinese symbols with other Chinese symbols. Searle argues that the man could not in fact be said to understand Chinese because to him the symbols have no meaning.

#### Social Implications of AI in Specific Issue Areas

The debate over AI's potential development and social impacts among these camps has been heated and, at times, acrimonious. The AI community in general tends to regard the disbelievers as wrong or irrelevant (14) and the social critics as largely too bitter and pessimistic to be productive (8). These arguments have become somewhat clichéd. However, the prospect for widespread and significant applications of AI has become more immediate in the mid-1980s, bringing new attention and new angles to the topic. Further, although AI is discontinuous from previous computer technologies in certain ways—e.g., in the use of symbols and knowledge bases rather than numbers and equations—its uses will be determined by

similar kinds of social and political factors that have governed society's use of a wide range of automated tools.

#### Economic Issues

Though there is a great deal of interest and writing in the economic implications of computer technologies, in this area it is difficult to separate the implications of AI from the implications of computers more generally. Several writers would propose, however, that there is in fact a qualitative difference between AI and other kinds of automation—namely, that although the net effect of automation has generally been to increase the number of jobs available through more productive industry and a healthier economy (albeit with some restructuring), intelligent computers may ultimately be able to perform such a wide range of tasks, both in manufacturing and in offices, that there will gradually be less and less need to employ as large a workforce as currently exists. Nilsson, e.g., essentially assuming the "maximum" scenario described above, writes (9):

*Quite simply, this hypothesis affirms that anything people can do, AI can do as well. Certainly AI has not yet achieved human-level performance in many important functions, but many AI scientists believe that artificial intelligence inevitably will equal and surpass human mental abilities—if not in twenty years, then surely in fifty. The main conclusion of this view of AI is that, even if AI does create more work, this work can also be performed by AI devices without necessarily implying more jobs for humans.*

Nilsson argues that society should give up the goal of full employment as being "unachievable, unnecessary and undesirable" and instead train people for human-service jobs (teaching, counseling, day care, health care) and in arts, crafts, literature, writing, and sports to better use their increasing leisure time. Nilsson's work builds on Boden's notion (15) that AI could be a "rehumanizing" rather than dehumanizing force for society. Noting that Polynesian cultures did not worry that abundance of mangos caused unemployment, she predicts:

*AI could be the Westerner's mango tree. Its contribution to our food, shelter, and manufactured goods, and to the running of our administrative bureaucracies can free us not only from drudgery but for humanity. It will lead to an increased number of "service" jobs—in the caring professions, education, craft, sport, and entertainment. Such jobs are human rather than inhuman, giving satisfaction not only to those for whom the service is provided, but also to those who provide it. And because even these jobs will very likely not be full-time, people both in and out of work will have time to devote to each other which today they do not enjoy. Friendship could become a living art again.*

As attractive as these somewhat utopian notions may be for those who are AI believers, several other arguments and factors are likely to make the future considerably more complex and less idyllic. First, although few analysts would dare make predictions beyond perhaps two decades from now, there is a large body of expert opinion to the effect that automation does not cause net job loss, regardless of the nature of the technology. Herbert Simon, for example, takes this view, asserting

that "Both standard economic analysis and a large body of empirical evidence demonstrate that there is no relation, positive or negative, between the technological sophistication of an economy and the level of employment it maintains" (16). That is, the economy is dynamic, and increased productivity in one place results in more demand and more employment in another; human ingenuity will always create new forms of employment in a healthy economy. James Albus, who can certainly be considered to be an AI believer, argues that intensive use of robots will result in "plenty of work for all able-bodied humans, plus as many robots as we could build" (17). Similarly, an executive director of the Joint Economic Committee of Congress states that the pace and magnitude of employment impacts for new technologies is generally overestimated (18).

A second factor to consider is the nature and quality of the jobs for those who work with automation. Although there is a general assumption that automation replaces onerous, tedious jobs and frees humans for creative tasks, this is not necessarily (and perhaps not even usually) the case. Robots have, in fact, greatly reduced the number of workers in dangerous and unpleasant spot-welding, spray-painting, and die-casting jobs. However, that is not the whole story. Many other boring jobs—such as janitors or fast-food service workers—have been and probably will continue to be uneconomical to automate. And case studies of the actual use of automation often show postautomation jobs that are machine-paced, monotonous, stressful, and/or unsatisfying because they have reduced necessary levels of human skill (19). In one case study of a robotized automobile welding plant, e.g., there had been a 150% annual turnover rate among first-line supervisors. One general foreman told a case-study researcher: "This has been the hardest 3 years of my life. There isn't any relaxation . . . I've walked out of here and sat in my car, unable to move, getting myself together" (19).

The amount of stress in a job is not just a function of the pace or demands of the job but also of the worker's range of decision-making latitude. Karasek, e.g., notes (20):

*The opportunity for a worker to use his skills and to make decisions about his work activity is associated with reduced symptoms (of stress) at every level of job demands. We do not find, therefore, support for the belief that most individuals "overburdened" with decisions face the most strain in an industrialized economy. Literature lamenting the stressful burden of executive decisionmaking misses the mark. Constraints on decisionmaking, not decisionmaking per se, are the major problem, and this problem affects not only executives but workers in low status jobs with little freedom for decisionmaking . . . e.g., assembly workers, garment stitchers, freight and materials handlers, nurse's aides and orderlies, and telephone operators.*

Potentially, AI could both constrain workers' decision-making latitude—since an intelligent machine can, to some extent, evaluate the options on its own—or it could enlarge that latitude by making workers feel more informed and powerful. Much depends on job and system design. There is clearly a wide variety of choice in using automation, and there is a considerable amount to learn in designing satisfying, reasonable jobs for those who work with automated technology. In addition, there will doubtless be some evolution of notions of what a satisfying and reasonable job entails as technology, industries, and society change. The Scandinavian countries have tended to pay more attention to such "work environ-

ment" issues than the United States and to ask whether jobs should be automated even if it is technically feasible to do so (19).

Most of the quantitative studies of the impact of computerized automation tend to indicate a significant, but considerably less than revolutionary, impact on the economy. The Upjohn Institute, e.g., has forecast displacement of 100,000–200,000 production-worker jobs due to robots by 1990 and creation of 10,000–20,000 jobs in robot maintenance and roughly 11,000 in robot-applications engineering (21). A frequently cited study by Ayres and Miller projected that so-called level I robots (roughly the technology on the market in 1981) could replace about 1.5 million metal-working craft workers, semiskilled machine operators, and laborers, whereas level II robots (next-generation machines with rudimentary sensing capabilities) could theoretically replace about 4 million out of the current total of 10.4 million of these workers (22). It should be cautioned that the Ayres and Miller study has frequently been quoted out of context. The time frame for this displacement is at least 20 years, and the estimates are the maximum, theoretically possible displacement. And as the researchers note, future generations of the technology will change the nature and magnitude of displacement. The study's figures are based on 16 responses to a survey of member firms of the Robot Institute of America (an association of robot vendors), who were asked to estimate the proportion of jobs within a given occupational title that could be done by the different levels of robots specified.

In contrast, Table 2 provides a qualitative picture of long-term trends in manufacturing occupations. These Bureau of Labor Statistics (BLS) data predict long-term declines in a wide range of manufacturing occupations except for engineers, adult-education teachers, computer-systems analysts, maintenance mechanics and repairers, and machinists. BLS data have, however, frequently been criticized for not paying sufficient attention to technological changes.

Generally, the robot "revolution" is proceeding much slower than expected in the early eighties, although the growth in use of robots is still substantial. A common estimate early in the decade was that the United States would have an installed base of 100,000 robots by 1990, from slightly over 6000 at the end of 1982. Events since then indicate that that estimate may be high by perhaps a factor of 2 (19).

There is little information available about the kinds of impacts other AI technologies, such as expert systems, will have on jobs. So far, although there are great expectations for use of expert systems, there are only a handful of actual uses. The late 1980s are a decisive time for this technology, as several hundred firms are attempting to develop expert-system products and in-house tools. Although some analysts speculate that this explosion will continue, others are concerned that expectations for expert systems are much too high and that there will be a backlash after many of these expert-system development projects fail to meet expectations (23). In any case, the effects of expert systems on employment levels and on the nature of work is a topic only beginning to receive attention.

From a more general economic view, many have argued that intensive use of automation—including AI—is inevitable in both offices and manufacturing and that competitive forces mean that society's choices are to "automate, emigrate, or evaporate." Although effective use of automation relative to other companies can be an important competitive tool, this argument is simplistic. Automation is often overrated as a sole

**Table 2. 1980 Employment for All Manufacturing Industries, Selected Automation-Sensitive Occupations<sup>a</sup>**

	Number	Percent	Long-Term Direction of Change
Engineers	579,677	2.85	+
Electrical	173,647	0.85	+
Industrial	71,442	0.35	+
Mechanical	122,328	0.60	+
Engineering and science technicians	439,852	2.16	+
Drafters	116,423	0.57	-
NC tool programmers	9,371	0.05	-
Computer programmers	58,622	0.29	-
Computer systems analysts	42,404	0.21	+
Adult education teachers	5,165	0.03	+
Managers, officials, and proprietors	1,195,743	5.87	?
Clerical workers	2,297,379	11.28	-
Production clerks	139,947	0.69	-
Craft and related workers	3,768,395	18.51	-
Electricians	126,001	0.62	+
Maintenance mechanics and repairers	391,524	1.92	+
Machinists, tool and die makers	356,435	1.75	-
Inspectors and testers	538,275	2.64	-
Operatives	8,845,318	43.44	-
Assemblers	1,661,150	8.16	-
Metalworking operatives	1,470,169	7.22	-
Welders and flamecutters	400,629	1.97	-
Production painters	106,178	0.52	-
Industrial truck operators	269,105	1.32	-
Nonfarm laborers	1,576,576	7.74	-
Helpers, trades	100,752	0.49	-
Stockhandlers, order fillers	104,208	0.51	-
Work distributors	16,895	0.08	-
Conveyor operators	31,469	0.15	-

<sup>a</sup> Bureau of Labor Statistics data collated in Ref. 19. Note: Data refer only to wage and salary workers.

source of economic competitiveness. Other factors are as important or more important, including especially the value of the dollar. Some firms have found that simply paying attention to improved, simpler design of their products, more economical use of materials, and streamlined layouts for their plants has a bigger payoff than installing robots (19). Finally, automation is not a panacea for competitive problems. One interesting recent theory asserts that although office automation helps well-managed organizations become more productive, it facilitates the decline of badly managed ones (24).

### Political and Control Issues

There are two important and rather distinct sets of issues related to AI and control over important social decisions. The first set is more general and concerns tendencies toward centralization or decentralization of power in society. The second set relates more specifically to control over military decisions.

**Centralization/Decentralization Issues.** This topic is very closely related to more general questions of the impact of computers on organizations, and hence one can gain some insight on AI's potential impact on centralization or decentralization from these more general studies. It would seem an elementary

observation that technology facilitates changes in the structures of organizations and society. It is by now almost a cliché to assert that information technology—i.e., computers and telecommunications—is a potent restructuring force because it can change the flow of information, and hence control, in work places and in society. Although there is certainly truth in this assertion, it has become clear over the past two decades that intuitively appealing notions of the nature of such impacts need to be carefully examined and tested in case studies. The impacts of information technology up to now, and the likely impacts of AI, are more subtle and complex than many have asserted.

For example, in a good summary of the literature on computing and organizations, Attewell and Rule (25) point out that a variety of analysts starting in the late fifties predicted that computing would lead to increased centralization of power in organizations by eliminating levels of middle management and allowing information to flow directly to top decision-makers. Some case-study research has confirmed that prediction by identifying relative declines in middle management and demonstrating that computer mail systems tend to be dominated by “top-down” communication. However, other studies found increased delegation of decision-making authority as a result of management information systems and increases in the complexity of organizational hierarchies. In sum, it appears that the context for introduction of computers has a decisive effect on the impacts and the ultimate organizational structure. Kling (26) provides insight into these contextual effects by pointing out the difference between a strictly “rationalist” view of the impacts of computerization and a view that accommodates more complex social dynamics:

*Systems rationalists typically emphasize the positive roles that computerized technologies play in social life. Often, they examine new capabilities of computing technologies (e.g., computer conferencing) or new areas in which existing computer technologies may be applied. They assume that there is a marked consensus on major social goals relevant to computing use, and they often develop a relatively synoptic account of social behavior. Systems rationalists place efficiency, whether economic or organizational, as the predominant value.*

*In contrast, segmented institutionalists examine the consequences of computerized technologies on many aspects of social life, both “legitimate” and “illegitimate.” For example, they observe that participants in organizations adopt computing to enhance their personal status or credibility, as well as to improve the technical quality of their decisions or increase the economic efficiency of specific activities. . . . They identify as dominant values the sovereignty of individuals and groups over critical aspects of their lives, the integrity of individuals, and social equity; economic or organizational efficiency is subservient to these values.*

Typically, versions of the more positive, more simplistic systems rationalist view are much more prominent than the segmented institutionalist view. Kling argues that much of computerization is thus pushed along by proponents of certain kinds of computerization who form loose coalitions called “computer-based social movements” (27). These movements, proponents of techniques ranging from urban information systems to office automation and personal computing, share certain ideological beliefs about computers (see Fig. 2).

For example, beginning in the 1980s the microcomputer has been hailed as a tool to decentralize the work place and,

1. Computer-based technologies are central for a reformed world.
2. The improvement of computer-based technologies will help reform society.
3. No one loses from computerization.
4. More computing is better than less, and there are no conceptual limits to the scope of appropriate computerization.
5. Perverse or undisciplined people are the main barriers to social reform through computing.

**Figure 2.** Key beliefs of participants in Kling's "computer-based social movements." (from Ref. 27).

indeed, society itself by empowering ordinary workers and citizens. Personal computers will bring "revolution, transformation and salvation" (1), unlimited opportunities for personal creativity, and, essentially, a new form of electronic democracy in which citizens are informed and participate in the political process via personal computer. Winner (28) and Carey and Quirk (29) point out that these predictions of electronic utopia are very similar to previous predictions of coming utopias as a result of steam or electricity. Carey and Quirk argue that these "exhortations" about the future positive effects of technology in essence tend to drown out other voices that point out that these technological trends have more complex and more mixed social effects. Winner, taking this notion one step further, tries to debunk the notion that lies at the heart of most utopian predictions involving personal computers—that "knowledge is power." Winner writes (28), "Surely there is no automatic, positive link between knowledge and power, especially if that means power in a social or political sense. At times, knowledge brings merely an enlightened impotence or paralysis."

AI is likely to have similarly complex effects on centralization and decentralization of power in society. Credible arguments can be made for important effects in both directions. For example, to the extent that powerful AI systems demand large-scale computers and special expertise for development, they are likely to be used only by large organizations that already wield substantial power—e.g., Western governments, wealthy universities, or hospitals. Weizenbaum (10), e.g., argued that the principal applications for speech-recognition systems would be for control of warships and government surveillance of the phone conversations of suspected subversives. (It should be noted that although such applications are indeed in the works, more rudimentary speech-recognition systems are now fairly cheap and usable in a wide variety of applications). Further, one application often mentioned for expert systems is to tie together a variety of large databases in a powerful way. Although such a distributed database manager has a variety of productive applications in a factory or office, it could also allow governments to combine disparate data about citizens into a de facto collection of frighteningly complete dossiers (30).

On the other hand, AI could be a powerful decentralizing force. In particular, AI could empower the citizen by making computers easier to use and adaptable to individuals. Intelligent computer-aided instruction (qv) could provide cheap, customized education. And some have suggested that AI systems could also help individuals cope with "information overload" because an intelligent personal computer could screen incoming information according to its owner's preferences. Finally, if expert systems were used to make social decisions, e.g., for eligibility in government programs, citizens could theoretic-

cally interrogate and scrutinize the rules and assumptions built into the system, thus making government more clear and accountable (7).

Although debates about social effects of technology are often polarized and it is thus tempting to assert that AI will either result in centralization or decentralization, it is likely that all of the effects described above—and others yet to be imagined—will occur in a fluid and complex society.

**Military Issues.** In questions of control over military decisions, the issue is not centralization but rather the extent to which AI may shift control over military decisions from people to machines and whether that shift of control is good or bad. At certain levels it is clear that the U.S. military indeed hopes to shift control from people to machines. For example, one of the primary goals of the DARPA's massive Strategic Computing Initiative (SCI) is to create an autonomous vehicle (qv) that could navigate through unfamiliar terrain and ultimately perhaps identify targets and fire at them (31). Other goals of the SCI involve a "pilot's associate" and a variety of battle-management programs that could help people cope with the massive amount of information involved in a high-technology battlefield. Defense officials have asserted that they do not intend to give a machine control over firing of nuclear weapons (32). On the other hand, there is a hope that AI systems will control nonnuclear systems of tremendous importance. The SCI notes, "An extremely stressing example . . . is the projected defense against strategic nuclear missiles, where systems must react so rapidly that it is likely that almost complete reliance will have to be placed on automated systems" (33).

A variety of critics, many of whom are involved in a new U.S. group called Computer Professionals for Social Responsibility, have severely criticized these uses of AI in the military. The critics have argued that the SCI monopolizes AI talent toward work with narrow military applications; that computing systems in general are not sufficiently reliable for life-or-death decisions, especially in "situations of extreme uncertainty"; and that more complicated AI-based weapons technology would simply fuel the arms race (34,35).

On the other hand, there is a minority in the AI community who argue that, in fact, maximum automation of warfare is desirable because ultimately intelligent computers are likely to make more rational, well-informed decisions and thus would avoid warfare (as well as other problems such as famine) (8). This view is presented here only for the sake of completeness; it not only assumes the "maximum" scenarios for development in AI but also assumes that society would make the moral choice to allow machines to take such control.

Clearly, the uses of AI systems in the military do raise significant moral issues that the military itself tends not to raise in its emphasis on the use of AI as just another tool. The fact that a machine might be created that is sophisticated enough to control an autonomous tank or a ballistic-missile defense system does not mean that the machine should be used to make such a decision. As Weizenbaum (10) forcefully argues:

*The question is not whether such a thing can be done, but whether it is appropriate to delegate this hitherto human function to a machine . . . however much intelligence computers may attain, now or in the future, theirs must always be an intelligence alien to genuine human problems and concerns. . . . What emerges as the most elementary insight is that, since*

*we do not now have any ways of making computers wise, we ought not now to give computers tasks that demand wisdom.*

Beyond these issues of moral responsibility, a final issue raised by possible transfers of control from people to machines is that of legal responsibility. On the one hand, attempting to hold a machine legally responsible for its actions seems an absurd thought. Searle argues that for something to be held responsible, it must be relevantly like a human being, with consciousness, autonomy, and responsibility. As discussed above, in his view appropriately programmed digital computers could never match this criterion because "formal simulations can never actually duplicate human mental processes" (36). Willick, on the other hand, notes that the law defines certain kinds of partial or limited persons (with limited rights), such as corporations, minors, and the insane. Consequently, he suggests that the law may eventually grant some machines some limited rights of personhood and responsibility (37). Although this argument is extraordinarily speculative, it is probably true that increasingly sophisticated computers will require gradual adjustment in legal concepts.

### Psychological and Sociological Issues

The continuing development and application of AI is certain to require psychological and conceptual adjustments on the part of ordinary citizens. Setting aside for the moment the question of under what conditions computers would deserve the term "intelligent" in a formal sense, it is clear that machines that increasingly *look* intelligent to a layperson will change relationships in society and will require changes in the way people think of themselves. Several writers have compared the potential psychological effects of AI to other scientific revolutions that have caused blows to the human psyche: Copernicus, who made people realize that the earth is not the center of the universe; Darwin, who informed humans that their ancestors are quite ordinary animals; and Freud, who presented people with the notion that they are not in control of much of their psyche. Thus, to the extent that intelligent machines appear to meet or exceed human intelligence in a broad sense—and, as noted in the section Future Directions, above, it is arguable to what extent this will occur—people are faced with the realization that their own intelligence is not unique and in some ways not supreme. People may be forced to reevaluate what they value in each others' intelligence and what it is to be human. And indeed, the need for psychological adjustment to intelligent machines could arrive much faster and become more immediate than adjustment to Copernicus, Darwin, or Freud because computers are encroaching upon people's daily lives at an increasing pace.

The magnitude of the psychological impact of AI depends in large part on people's willingness to ascribe intelligence to the machine. Thus, one can gain some insight about these potential impacts by examining the concept of intelligence and how it is used, both by laypersons and experts. Developing a formal definition of intelligence (qv) has proved to be an elusive endeavor for the field of psychology for decades. There has of course been great controversy, e.g., about just what intelligence tests are measuring and whether they are socially productive (38). The most widely accepted definition is that intelligence involves the "ability to overcome difficulties in new situations" (39), although that notion is surely unsatisfying in its lack of detail and explanatory power. There has, however,

been more success in identifying the characteristics that people use informally in judging intelligence in others. That is, despite the fact that a formal definition of intelligence is troublesome, people appear to have a reasonable, loosely organized consensus about the kinds of behaviors that are components of intelligence.

The work of Sternberg, Conway, Ketron, and Bernstein (40) is most instructive in dissecting people's intuitive notions of intelligence—what the researchers call "implicit theories of intelligence." The characteristics of a prototypical intelligent person from the work of Sternberg's group are outlined in Table 3. The Yale researchers asked subjects to list the charac-

**Table 3. Factors Underlying People's Conceptions of Intelligence: Laypersons Rating Characteristicness in Ideal Person<sup>a</sup>**

Factor	Factor Loading
I. Practical problem-solving ability:	
Reasons logically and well	0.77
Identifies connections among ideas	0.77
Sees all aspects of a problem	0.76
Keeps an open mind	0.73
Responds thoughtfully to others' ideas	0.70
Sizes up situations well	0.69
Gets to the heart of problems	0.69
Interprets information accurately	0.66
Makes good decisions	0.65
Goes to original sources for basic information	0.64
Poses problems in an optimal way	0.62
Is a good source of ideas	0.62
Perceives implied assumptions and conclusions	0.62
Listens to all sides of an argument	0.61
Deals with problems resourcefully	0.61
II. Verbal ability:	
Speaks clearly and articulately	0.83
Is verbally fluent	0.82
Converses well	0.76
Is knowledgeable about a particular field of knowledge	0.74
Studies hard	0.70
Reads with high comprehension	0.70
Reads widely	0.69
Deals effectively with people	0.68
Writes without difficulty	0.65
Sets aside time for reading	0.64
Displays a good vocabulary	0.61
Accepts social norms	0.61
Tries new things	0.60
III. Social competence:	
Accepts others for what they are	0.88
Admits mistakes	0.74
Displays interest in the world at large	0.72
Is on time for appointments	0.71
Has social conscience	0.70
Thinks before speaking and doing	0.70
Displays curiosity	0.68
Does not make snap judgments	0.68
Makes fair judgments	0.66
Assesses well the relevance of information to a problem at hand	0.66
Is sensitive to other people's needs and desires	0.65
Is frank and honest with self and others	0.64
Displays interest in the immediate environment	0.64

<sup>a</sup> Reference 40.

teristics of intelligence; then another group of subjects rated those attributes for characteristicness in an ideal intelligent person. Practical problem-solving ability, verbal ability, and social competence surfaced as three general categories of criteria, with more specific abilities outlined for each category. After a similar study on experts' conceptions of intelligence, Sternberg's group concluded that experts and laypersons "see things very much the same way, although not identically." In addition, the researchers found that intelligence prototypes were stable and reliably similar across subjects and that people indeed use their prototypes in judging the intelligence of themselves and others. The Sternberg study is consistent with other research on intelligence prototypes (41).

A suggestive experiment (42) was conducted in order to determine the extent to which people were willing to ascribe some of the characteristics of intelligence identified in the Sternberg study to machines. Subjects (laypersons of mixed ages and educational backgrounds) were asked to rate the ability of a machine to display some of these characteristics as well as other key traits. As might be expected, subjects rated the computer highest on characteristics under the factor "practical problem-solving ability," mid-range on "verbal ability," and low on "social competence." Computers received fairly high marks for intelligence (mean 3.9 on a scale of 1 = low to 7 = high), compared to humans (mean 5.5). Indeed, the score for computer intelligence in 20 years is virtually identical to that for humans (mean 5.4). Thus, these subjects apparently believed that AI will be largely successful. Moreover, subjects tended to believe that AI has *already* been far more successful than it has been. Subjects rated the computer quite high on identifying connections among ideas, seeing all aspects of a problem, and fairly high on considering the end result of actions and being verbally fluent. As AI researchers can painfully testify, these capabilities are among the most difficult and distant AI goals. This general trend is supported in other studies; in a survey of a college statistics class in 1981 (42), 54% of the class said a computer could beat the world chess champion now (see Computer chess methods), and only 23% felt this feat could not be achieved in 5 years. In another survey reported in Boden (43), a third of the subjects in 1974 believed a computer program existed that could beat Bobby Fisher. As Boden notes, this shows a "faith in the achievements of artificial intelligence that is decidedly misplaced."

Other empirical research, also tentative, suggests that this faith in the computer is accompanied by an unusual willingness to believe computer output. A study by Reys (44), e.g., asked subjects who were selected for mathematical estimation ability to estimate the answers to seven math problems and then use a calculator to check the accuracy of their answers. Reys, however, had programmed the calculator to make systematically increasing errors in computing, ranging from 10 to 50% higher than the actual answer. Only 20% of the participants (all male) voiced any suspicion after the first problem. Even after all seven problems 23% of the males and 64% of the females still had not expressed any doubt as to the accuracy of the calculator. In later comments they blamed themselves instead for not guessing closely enough. In another suggestive experiment (42) subjects were presented with questions and answers for four different kinds of questions—mathematical, scientific, factual social-science, and social-values topics. Although the answers to the questions were always the same,

the source for the answer was varied randomly as either a public-opinion poll, a newspaper or magazine, a textbook or other human-expert opinion or a "computer-based logic-reasoning program." When subjects were asked to score their level of agreement with the answers, they gave the computer roughly as much credibility as human experts or texts, with the exception of a slightly higher score for text/experts in factual social-science questions. Even for questions dealing with values, subjects gave answers from a computer-reasoning program ratings comparable to text/experts, media, and polls.

Although these studies are really only sketches of the kinds of research that could be performed on the psychological implications of AI, they suggest that people already have significant misconceptions about computers. These misconceptions may be the beginning of a very difficult conceptual-adjustment process as computers and AI become more pervasive. In a worst-case scenario, many of those that have the most misconceptions about AI may also be the ones displaced by automation technology (older, less-skilled workers), resulting in a significant alienation problem. However, such a scenario is purely speculative, and further research is needed to assess the psychological issues related to society's use of AI. The problem of psychological and sociological adjustment is doubtless much more significant than it may appear on first glance; for even if AI and other technology provide tremendous advances in productivity, those advances could be fruitless for many if they come at the cost of significant alienation and confusion. Davis (45) puts the dilemma eloquently:

*My belief based on the experience of us all is that the balance of power and the ratio of intelligence between man and computer is still indeterminate. Further, it is not entirely under man's control. In particular, as computers increase their capacities to perform more of the tasks formerly considered only within man's intellectual province, man must have and finally has, the opportunity to equip himself for other functions and a superior existence, or else his survival will seem less important to himself, leading quite unnecessarily to a physical and intellectual ennui.*

If one assumes that in fact AI systems will become powerful enough to give people this choice between superior existence or ennui, the path chosen—and the people capable of and empowered to make that choice—becomes a profound social issue.

### Technical Spin-off Issues

AI has begun to affect other sciences in a variety of ways—as a tool both for helping to conduct or analyze research and for training or advising practitioners. Although a complete analysis of these effects is beyond the scope of this entry, many of these impacts are described in other entries in this encyclopedia. AI's applications as a research tool can be seen most vividly in the use of Dendral by the chemistry community (see Dendral; Chemistry, AI in). A variety of medical applications use AI systems in training of doctors or, ultimately, as advisors to doctors (see Caduceus; Medical-advice systems).

AI has had, and will continue to have, a particularly strong impact on psychology (see Cognitive psychology). In this field AI has become not only a research tool but also a metaphor;



some researchers are phrasing psychological theories in terms related to computers and AI languages and theories, at times testing those theories by constructing programs. The potential usefulness of this marriage of disciplines has been a strong impetus to the growth of cognitive science (qv), which combines expertise and techniques from psychology, philosophy, linguistics, and computer science. Although some believe that using a machine to model the mind is a productive path for research, others argue that it is sterile and restrictive to attempt to understand the mind as a machine (Ref. 43 has a good summary of this debate; see also Cognitive science; Cognitive psychology).

Finally, as noted earlier, many researchers believe that AI will facilitate major changes in the field of education by allowing relatively inexpensive, customized training programs that can be delivered in a wide variety of settings (see Education applications; Computer-aided instruction, intelligent).

### Concluding Remarks

In any analysis of the potential social implications of technology, one is faced with two troubling questions: "To what extent can the direction of the technology and its impacts be predicted with any reliability?" and "If negative impacts are identified, can or should the direction of the technology be changed, or must society simply adjust?"

A realistic answer to the first question, as noted in the section Future Directions at the start of this entry, is that predictions are extremely limited in reliability, especially in a young (perhaps adolescent) field such as AI. Barring explosive advances in theory, the field will probably continue to move rather more modestly than its adherents would prefer. And because it takes time for AI to be applied and for its impacts to be felt, the social implications will unfold over the course of the next three or four decades. Nevertheless, if the field succeeds at only a portion of the problems it tackles, the long-term implications are likely to be profound, and thus the need to understand them is great. Snow, who wrote the seminal essay on the gulf between scientists and nonscientists (46), offers this mandate (47):

*The computer revolution is going to be the biggest technological revolution men have ever known, far more intimately affecting men's daily lives, and, of course, far quicker than either the agricultural revolution of neolithic times or the early industrial revolution. . . . We shall also need many people of different abilities who are at every step of the way studying, controlling and humanising [the computer revolution's] effects. . . . There was some excuse for our ancestors not foreseeing the effects of the first industrial revolution. There is no excuse this time.*

Snow's mandate clearly relates to the second question and brings up the fact that society—particularly in the United States—has not chosen to "study, control and humanize" the use of computers to a significant extent. Winner (48) provides an extremely articulate description of society's unwillingness to act on its choices regarding technology use and the consequences of that situation. Countries in Western Europe have taken somewhat more active measures, such as privacy-protection boards and standards and research related to the work environment of those who use computers (19). It is important

to recognize that there are in fact many social choices that influence the direction of AI and its effects, ranging from the way in which the government funds research [e.g., the balance between military and civilian research funding (49)] to the choices employers make in using technology and designing jobs (50) and the choices researchers make in selecting directions for their work (51). The extent to which citizens explicitly recognize and act upon these choices seems closely linked to the chances for realization of the more positive visions of AI's social effects.

### Reading List

Fleck (3) contains an interesting analysis of the sociology of AI, and elaboration of the various attitudes toward thinking machines.

Dreyfus and Weizenbaum (10,11) are by now classics in analysis of AI's social impacts, although the authors have distinct personal views about the topic. Reference 12 updates Dreyfus's earlier work and promises to be just as controversial. Boden (43) provides a more balanced analysis. Winegrad and Flores (53) is a careful (albeit clearly skeptical) analysis of the prospects for AI's development.

The proceedings of the Ninth International Joint Conference on Artificial Intelligence (see Ref. 37, pp. 1254–1309) contains a selection of short articles on related topics.

For broader views of the social impacts of computing, see Refs. 26 and 52.

### BIBLIOGRAPHY

1. E. A. Feigenbaum and P. McCorduck, *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*, Addison-Wesley, Reading, MA, 1983.
2. W. Stockton, "Creating computers that think," *New York Times Magazine*, December 7, 1980, pp. 40–43; and "Computers that think," *New York Times Magazine*, December 13, 1980, pp. 48–54.
3. J. Fleck, Artificial Intelligence and Industrial Robots: An Automatic End for Utopian Thought? in E. Mendelsohn and H. Nowotny (eds.), *Sociology of the Sciences*, Vol. 8, *Nineteen Eighty-Four: Science between Utopia and Dystopia*, Reidel, Dordrecht, pp. 189–231, 1984.
4. Office of Technology Assessment, U.S. Congress, *Artificial Intelligence: A Background Paper*, U.S. Government Printing Office, Washington, DC, 1987.
5. H. A. Simon and A. Newell, "Heuristic problem solving: The next advance in operations research," *Operat. Res.* **6**, 1–10 (January/February 1958).
6. M. Minsky, "Why people think computers can't," *AI Mag.* **3**, 3–15 (Fall 1982).
7. J. Fox, Judgement, Policy and the Harmony Machine, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 18–23, 1985.
8. P. McCorduck, *Machines Who Think*, Freeman, San Francisco, CA, 1979.
9. N. Nilsson, "Artificial intelligence, employment, and income," *AI Mag.* **5**, 5–14 (Summer 1984).
10. J. Weizenbaum, *Computer Power and Human Reason: From Judgment to Calculation*, Freeman, San Francisco, CA, 1976.
11. H. L. Dreyfus, *What Computers Can't Do: The Limits of Artificial Intelligence*, Harper & Row, New York, 1979.

12. H. L. and S. Dreyfus, "Why computers may never think like people," *Technol. Rev.* **89**, 42-61 (January 1986); and H. L. and S. Dreyfus, *Mind Over Machine*, MacMillan/Free Press, New York, 1986.
13. J. Searle, "Minds, brains, and programs," *Behav. Brain Sci.* **3**, 417-424 (1980).
14. S. Papert, *The Artificial Intelligence of Hubert L. Dreyfus: A Budget of Fallacies*, AI Memo 154, MIT AI Laboratory, Cambridge, MA, 1968.
15. M. A. Boden, Artificial Intelligence as a Humanizing Force, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, 1983, pp. 1197-1198.
16. H. Simon, "What computers mean for man and society," *Science* **210**, 1186-1191 (1977).
17. J. Albus, "The robot revolution: An interview with James Albus," *ACM* **26**, 179-180 (March 1983).
18. B. Bartlett, "The Luddite Answer to Unemployment," *Wall Street Journal*, p. 29, July 18, 1983.
19. Office of Technology Assessment, U.S. Congress, *Computerized Manufacturing Automation: Employment, Education, and the Workplace*, U.S. Government Printing Office, Washington, DC, 1984.
20. R. A. Karasek, Jr., "Job demands, job decision latitude, and mental strain: Implications for job redesign," *Admin. Sci. Quart.* **24**, 303 (June 1979).
21. H. A. Hunt and T. L. Hunt, *Human Resource Implications of Robotics*, Upjohn Institute for Employment Research, Kalamazoo, MI, 1983.
22. R. U. Ayres and S. M. Miller, "Robotics and conservation of human resources," *Technol. Soc.* **4**, 181-198 (1982).
23. D. McDermott, M. M. Waldrop, R. Schank, B. Chandrasekaran, and J. McDermott, "Panel discussion: Dark ages of AI," *AI Mag.* **6**, 122-134 (Fall 1985).
24. P. A. Strassman, *Information Payoff; The Transformation of Work in the Electronic Age*, Free Press, New York, 1985.
25. P. Attewell and J. Rule, "Computing and organizations: What we know and what we don't know," *ACM* **27**, 1184-1192 (December 1984).
26. R. Kling, "Social analyses of computing: Theoretical perspectives in recent empirical research," *Comput. Surv.* **12**, 61-110 (March 1980).
27. R. Kling and S. Iacono, Computerization as the Product of Social Movements, in R. Gordon (ed.), *Microelectronics in Transition*, Ablex, Norwood, NJ, 1987.
28. L. Winner, "Mythinformation in the high-tech era," *IEEE Spectr.* **21**, 90-96 (June 1984).
29. J. W. Carey and J. J. Quirk, The History of the Future, in G. Gerbner, L. P. Gross, and W. H. Melody (eds.), *Communications Technology and Social Policy*, Wiley, New York, pp. 485-503, 1973.
30. Office of Technology Assessment, U.S. Congress, *Federal Government Information Technology: Electronic Record Systems and Individual Privacy*, U.S. Government Printing Office, Washington, DC, June 1986.
31. Defense Advanced Research Projects Agency, Strategic Computing: First Annual Report, Washington, DC, February, 1985.
32. R. S. Cooper, "Strategic computing initiative: An exchange," *Bull. Atom. Sci.* **41**, 54-56 (January 1985).
33. Defense Advanced Research Projects Agency, *Strategic Computing*, Department of Defense, U.S. Government Printing Office, DC, 1983.
34. S. M. Ornstein, B. C. Smith, and L. A. Suchman, "Strategic Computing," *Bull. Atom. Sci.* **40**, 11-15 (December 1984).
35. A. Borning, Computer System Reliability and Nuclear War, unpublished paper, Computer Scientists for Social Responsibility, Palo Alto, CA, August 1985.
36. J. R. Searle, Computers, the Mind, and Responsibility, paper prepared for Workshop on Long-Term Social Implications of Machine Intelligence, Office of Technology Assessment, U.S. Congress, November 9, 1984.
37. M. S. Willick, Constitutional Law and Artificial Intelligence: The Potential Legal Recognition of Computers as "Persons," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, pp. 1271-1273, 1985.
38. S. J. Gould, *The Mismeasure of Man*, Norton, New York, 1981.
39. H. J. Eysenck, W. Arrol, and R. Melli (eds.), *Encyclopedia of Psychology*, Herder and Herder, New York, 1972.
40. R. J. Sternberg, B. E. Conway, J. L. Ketron, and M. Bernstein, "People's conceptions of intelligence," *J. Personal. Soc. Psychol.* **41**, 37-55 (1981).
41. U. Neisser, The Concept of Intelligence, in R. J. Sternberg and D. K. Detterman (eds.), *Human Intelligence: Perspectives on its Theory and Measurement*, Ablex, Norwood, NJ, 1979.
42. J. A. Dray, Coping with the "Thinking" Machine, unpublished thesis, Wesleyan University, 1982.
43. M. A. Boden, *Artificial Intelligence and Natural Man*, Basic Books, New York, 1977.
44. L. Timnick, "Electronic bullies," *Psychol. Tod.* **16**, 10-15 (February 1982).
45. R. M. Davis, "Evolution of computers and computing," *Science* **195**, 1187-1191 (March 11, 1977).
46. C. P. Snow, *The Two Cultures and A Second Look*, Cambridge University Press, Cambridge, U.K., 1964.
47. C. P. Snow, "Government, science, and public policy," *Science* **151**, 650-653 (February 11, 1966).
48. L. Winner, *Autonomous Technology*, MIT Press, Cambridge, MA, 1977.
49. Office of Technology Assessment, U.S. Congress, *Information Technology Research and Development: Critical Trends and Issues*, U.S. Government Printing Office, Washington, DC, 1985.
50. C. Pava, *Managing New Office Technology: An Organizational Strategy*, Free Press, New York, 1983.
51. T. Winograd, "Some Thoughts on Military Funding," *CPSR Newslett.* **2**, 1-3 (Spring 1984).
52. R. Kling, "Value conflicts in computing developments," *Telecommun. Pol.* **12**-34 (March 1983).
53. T. Winograd and F. Flores, *Understanding Computers and Cognition*, Ablex, Norwood, NJ, 1986.

J. DRAY  
Office of Technology Assessment  
U.S. Congress

## SOCIETIES, AI

Groups of people interested in artificial intelligence first gathered together in the context of other societies, particularly those in computer science and electrical engineering. AISB and SIGART (see Table 1) were probably the first formal organizations, although the chronology is not yet clear. In recent years the number of associations has proliferated locally, nationally, and regionally. Many of them publish journals and sponsor symposia (see Literature of AI for descriptions of some

of these publications). As yet, there is no international member organization, although IJCAI coordinates artificial intelligence activities globally through its biennial conferences. Table 1 provides a list of a number of the AI societies currently active, together with some of the parent organizations and related groups that support work in the field.

**Table 1. AI Societies**

AAAI	American Association for Artificial Intelligence 445 Burgess Drive Menlo Park, CA 94025 415-328-3123	CSCSI/ SCEIO	Canadian Society for Computational Studies of Intelligence/Societe Canadienne pour l'Etude de l'Intelligence par Ordinateur c/o Canadian Information Processing Society 243 College Street, 5th floor Toronto, Ontario, CANADA M5T 2Y1 416-593-4040
AAR	Association for Automated Reasoning c/o William McCune MCS-203 Argonne National Laboratory Argonne, IL 60439-4844 312-972-3065	CSS	Cognitive Science Society c/o Kurt van Lehn Psychology Dpt., Carnegie-Mellon University Pittsburgh, PA 15213 412-268-4964
ACH	Association for Computers and the Humanities c/o Dr. Donald Ross English Dpt., University of Minnesota Minneapolis, MN 55455 612-625-2888	IEEE	IEEE Computer Society 10662 Los Vaqueros Circle Los Alamitos, CA 90720-2578 714-821-8380
ACL	Association for Computational Linguistics c/o Donald E. Walker Bell Communications Research 435 South Street, MRE 2A379 Morristown, NJ 07960-1961 201-829-4312	IFSA	International Fuzzy Systems Association c/o Hans-Juergen Zimmermann Aachen Institute of Technology D-5100 Aachen, FRG 49-241-806182
ACM	Association for Computing Machinery 11 West 42nd Street, 3rd Floor New York, NY 10036 212-869-7440	IJCAI	International Joint Conferences on Artificial Intelligence, Inc. c/o Donald E. Walker (see ACL)
AFCET	Association Francaise pour la Cybernetique, Economique, et Technique c/o Yves Kodratoff Universite de Paris Sud Bat. 490 F-91405 Orsay Cedex, FRANCE 33-1-6941-6750	NACCC	North American Computer Chess Championship c/o Hans Berliner Carnegie-Mellon University Pittsburgh, PA 15213 412-268-2577
AFIPS	American Federation of Information Processing Societies 1899 Preston White Drive Reston, VA 22091 703-620-8900	NAFIPS	North American Fuzzy Information Processing Society c/o Ralph Shear 1916 Bayberry Road Edgewood, MD 21040 301-679-4673
AISB	Society for Artificial Intelligence and Simulation of Behavior c/o Richard Young MRC Applied Psychology Unit Cambridge, CB2 2EF, UK 44-223-355294	RIA	Robot Institute of America PO Box 930 Dearborn, MI 48128 313-271-0778
ALLC	Association for Literary and Linguistic Computing c/o Thomas Corns English Dpt., University College of North Wales Bangor, Gwynedd LL57 2DG, UK 44-248-351151	SCS	Society for Computer Simulation c/o Charles A. Pratt PO Box 17900 San Diego, CA 92117 619-277-3888
AMS	American Mathematical Society P.O. Box 6248 Providence, RI 02940 401-272-9500	SIGART	Special Interest Group of the ACM on Artificial Intelligence
ARC	Association pour la Recherche Cognitive c/o Daniel Kayser Universite de Paris Sud Bat. 490 F-91405 Orsay Cedex, FRANCE 33-1-6941-6750	SIGMOD	Special Interest Group of the ACM on Management of Data
		SIGPLAN	Special Interest Group of the ACM on Programming Languages ACM Headquarters (see ACM)
		SME	Society of Manufacturing Engineers SME Drive PO Box 950 Dearborn, MI 48128 313-271-1500
		SPIE	Society of Photo-Optical Instrumentation Engineers PO Box 10 1022 19th Street Bellingham, WA 98225 206-676-3290

## SOPHIE

SOPHIE is a tutorial system that aids a student in troubleshooting malfunctioning equipment in an electronics labora-

tory and was written from 1973 to 1978 by Brown, Burton, and others at Bolt, Beranek and Newman, Inc. in Cambridge, Massachusetts (see J. S. Brown, R. R. Burton and J. de Kleer, Pedagogical, Natural Language and Knowledge Engineering Techniques in SOPHIE I, II, and III, in D. Sleeman and J. S. Brown (eds.), *Intelligent Tutoring Systems*, Academic Press, New York, pp. 227–282, 1982).

M. R. TAIE  
SUNY at Buffalo

## SPEECH ACTS

Speech-act theory is a theory of communication developed by philosophers that is readily adapted into computational models of language. The fundamental premise is that an utterance, i.e., the speaking of a sentence, consists of two major components: its semantic content and the intentions of the speaker. For example, essentially the same semantic content is used in different communicative situations in the following sentences:

I drove my boat onto the rocks.  
Did I drive my boat onto the rocks?  
Drive my boat onto the rocks!

The common semantic part in each of these sentences is termed “propositional content.” The difference in use, i.e., uttering an assertion, question, or command, respectively, is called the “illocutionary force” of the sentence. Speech-act theory provides an attractive theory of the force of sentences based on a general theory of action.

The origins of this approach lie in philosophy. Austin first observed that some sentences could not be analyzed in terms of their truth conditions (1). For example, it makes no sense to talk of an utterance such as “I hereby sentence you to 10 years hard labor” as being true or false, although one may talk about whether a sentence was uttered appropriately and therefore was a valid sentencing. For instance, in the courtroom, when the judge says this utterance at the appropriate time, the judge is performing the act of sentencing the prisoner. If uttered by someone other than the judge, it has no such effect (i.e., force).

This same argument holds for other sentences as well. For example, a promise cannot be said to be true or false, but rather it is said to be sincere or not, depending on whether the speaker really intends to do the act promised. Similarly, a question cannot be judged as true or false. Again, a question is sincere if the speaker wants to know the answer. Even assertions cannot be judged to be true or false. The proposition in the assertion may be true or false, and the assertion is sincere if the speaker believes that the proposition asserted is true. Thus, a comprehensive analysis of the use of sentences is better couched in terms of sincerity conditions on the utterance.

There have been many attempts to categorize the range of speech acts on the basis of the type of sincerity conditions that must hold. For instance, Searle suggests five classes of speech acts. The representatives include the making of “assertions,” “claims,” “denials,” etc. All members of this class commit the speaker to some degree of belief that the propositional content holds. The prototypical member discussed below is “inform.”

The directives include “commands,” “pleas,” “questions,” etc., and all commit the speaker to an attempt to get the hearer to do something. The prototypical member of this class is “request.” The commissives all commit the speaker to some degree of commitment to perform some act. The prototypical member is “promise.” Expressives all commit the speaker to some psychological state and include “thanking,” “apologizing,” and “congratulating,” among many others. Finally, the declarations include the first class of acts that Austin identified, where a particular act is performed by explicitly using the appropriate verbs. This includes telling someone “You’re fired,” the umpire calling someone out in baseball, sentencing a prisoner, etc.

## Direct Mapping Models of Speech Acts

Many natural-language systems can be viewed as using some form of speech-act theory to characterize the intent of the utterances. These systems recognize the appropriate speech act from a set of features of the sentence. This is called the direct mapping approach. For example, an imperative-mood sentence would be a request, an interrogative would be a question, etc. With this categorization, various properties and consequences of the utterance can be conveniently encoded. For example, if an INFORM is defined to be an action with an effect that the hearer knows the propositional content and a prerequisite that the speaker knows the propositional content, then if the system receives the utterance “The ball is red” from the user and recognizes this as an INFORM, it should conclude that the user wants the system to believe the ball is red (from the effect), and the user believes the ball is red (from the prerequisite). In other words, since an INFORM was done, the prerequisite must be true and the user must want the effect.

In a similar manner, if the system recognizes “Is the ball red?” as a question, i.e., a request that the system inform the user of the answer, the system should conclude that the user wants the system to tell the user whether the ball is red. This would be encoded as the effect of the REQUEST act.

The situation becomes more complicated when considering indirect speech acts (ISAs), which are sentences that literally mean one thing but are used in some other way. For example, the sentence “Can you pass the salt?” literally is a yes/no question about the hearer’s abilities, yet in most contexts it is a request that the hearer pass the salt. In the direct mapping approach, the analysis of “can you” sentences as REQUESTs results from the specific syntactic form of the sentence. In this case the system might have a rule that all sentences of the form “Can you do something” are requests to do that thing.

The problem with this approach in general, of course, is that such sentences can also be used literally in certain contexts. For example, “Can you lift that weight?” might be a simple yes/no question in many contexts. Even “Can you pass the salt?” could be meant literally as a yes/no question in a context where the hearer’s abilities were in question. The direct mapping approach can be augmented to include tests on the context. For example, the following test might be used for sentences of the form “Can you X”: If the hearer can do the mentioned action, treat as a request to do X; otherwise, treat as a yes/no question. These techniques, however, are generally limited in their effectiveness and are ad hoc. Furthermore, there are large classes of indirect speech acts that seem not to be handled by the approach. A classic example is the utterance “It’s cold in here,” said by a king to his servant and

meant as a request to close the window. There are no simple tests on the world given such a sentence that could derive this interpretation. Rather, it requires extensive world knowledge about actions, causality, and peoples' intentions.

### Plan-Based Models of Speech Acts

In these models, speech acts are embedded in a theory of action used for planning (qv). Thus, speech acts are put on the same footing as physical actions, such as moving blocks, and a single inference (qv) technique can be used to reason about both the context of the utterance and the utterance itself.

For example, consider a simple world with one room with an open doorway. The class of actions consisting of going outside might be defined in a STRIPS-like formalism (2) as follows:

GO-OUTSIDE(robot)

preconditions: robot is INSIDE

robot wants to GO-OUTSIDE(robot)

effect: robot is OUTSIDE

Thus, if the robot is inside and wants to go outside, the robot may execute the action GO-OUTSIDE with the effect that the robot will now be outside. The key step here is the want-precondition that asserts that the robot wants to do the act. With this condition it can be assumed that if all of the preconditions of an act by another agent are satisfied, including the want precondition, the agent can be expected to do that act.

Similarly, a speech act like REQUEST might be defined as follows (ignoring many technicalities):

REQUEST(speaker, hearer, act)

precondition: speaker WANT hearer to do ACT

effect: hearer WANT to do ACT

Consider constructing a simple plan to get a robot R to go outside. In order for this to occur, the planner must make sure the preconditions hold. Assuming that R is inside currently, one needs only to get R to want to go outside. This can be achieved by performing a REQUEST act. Since the goal was to get R to go outside, the precondition for the REQUEST is already true. Thus, there is an executable plan: Do a REQUEST to R that R go outside, which has the effect that R wants to go outside, which satisfies all of the preconditions for R actually performing the action, and thus R will go outside. Although in an actual analysis many more issues need to be considered, this example presents the essential points in linking speech acts to planning systems.

This framework yields several implications as a model of the language-generation (see Natural-language generation) process. For example, consider the situation where the doorway was closed and a key was needed to open it. In planning to get R to go outside, one precondition on going outside, namely that the door is open, would be unsatisfied. This problem could be resolved by getting R to open the door by first getting the key and, if R does not know where the key is, by planning to INFORM R of the key's location. Thus, one might plan several speech acts in order to accomplish a single goal.

Of course, nothing has been said so far as to how the speech acts can be realized by utterances. The simplest approach is to use something like the direct mapping approach to generate a sentence for each speech act. A better approach is to enrich the action vocabulary and devise a planning model that can generate a single utterance that accomplishes several speech acts simultaneously. For example, in the situation with the key above, one might perform both the REQUEST to go outside and the INFORM that the key is in the corner with the utterance "Go outside using the key in the corner."

The plan-based theory is also useful in language understanding (see Natural-language understanding). Assuming that the model of generation above is plausible, it seems that one of the most important things to consider when analyzing someone else's utterance is what plan one had that produced the utterance. In other words, the planning procedure can be reversed (plan recognition) to analyze the intentions underlying an utterance by another agent. Even if the speaker's utterances were always direct and explicit, this step would still be necessary since in many conversations answers that go beyond the explicit request are necessary. On top of this, speakers are often not explicit in their requests. For example, the following are actual dialogues collected at an information booth in a train station between a passenger (P) at the station and the clerk (C) in the booth:

P: When is the Windsor train?

C: 3:15 at Gate 7.

Although P spoke directly, the clerk's response included additional information. The clerk knows that the gate is relevant because he has identified P's plan to board the train to Windsor from the initial utterance. This is called a helpful response. Consider another example:

P: The 3:15 to Windsor?

C: Gate 7.

In this case P does not even utter a complete sentence, yet the single noun phrase (NP) provides enough information that the clerk can recognize P's intentions and respond appropriately.

Both these cases can be explained using the plan-recognition theory. In each case the utterance is taken at face value as a speech act, and from that act, and knowledge of expected high-level goals in the train-station setting, a plan can be recognized. The clerk's response is then based on what would be helpful for P to execute the recognized plan rather than on the surface form of the initial utterance.

This approach is useful for analyzing indirect speech acts as well by taking the initial utterance literally and recognizing a plan. For an utterance such as "Can you tell when the Windsor train leaves?" the recognizer might perform the following line of reasoning:

1. P asked whether I could tell him when the Windsor train leaves.
2. Thus, P probably wants to know if I know when that train leaves (the effect of 1).
3. But given that P knows I'm an information clerk, I expect P to know already that I do know when the train leaves.
4. Thus, P must be interested in something relating to my knowledge of departure times.

5. Since my knowing the departure time is a necessary prerequisite of my informing P of the time, it is possible that P wants me to inform him of the departure time.

This can all be expressed as a line of plausible inference (see Reasoning, plausible) using only general knowledge about planning and intentions, though there is not the space to describe it in detail here. The point is that by assuming a plan-recognition model and a helpful conversant, many forms of indirect speech acts seem to be interpretable by starting with their literal interpretation.

### Indirect Speech Acts and the Recognition of Intention

What the above account of indirect speech acts does not do is distinguish between a mere helpful response and a true indirect speech act. For example, it cannot distinguish between the case where I say "There's no chalk here" and you, being helpful, get some chalk for me, and the case where I say "There's no chalk here" and mean it to be a request that you get some chalk. The difference is brought into focus by considering the situation where you do not go to find some chalk. In the helpful response case you may simply do nothing, and all is fine. In the indirect speech-act case you would be obliged to offer some apology or refusal if you do not get the chalk. To capture these distinctions, the nature of communication must be considered in more detail.

Grice pointed out that what distinguishes acts of communication from other physical acts is that they are performed not only with the intention to produce an effect on the hearer but also with the intention that the hearer recognize the speaker's intention (3). For example, for me to say "There's no chalk here" and mean it as a request that you go and get some chalk, I must intend that you recognize that I said "There's no chalk here" in order to ask you to get the chalk. Thus, to recognize speech acts, the hearer must not only recognize the speaker's plan but also recognize what plan the speaker intended the hearer to recognize.

In fact, it can be shown that in order for communication to succeed, this plan that the speaker intends the hearer to recognize must become part of the shared knowledge (or mutual beliefs) of the speaker and hearer. In other words, the hearer believes the speaker believes it, and the hearer believes the speaker believes the hearer believes it, etc., to arbitrary levels of nesting of belief (see Belief systems).

This suggests the following extended model of analysis of utterances: The sentence is taken literally, and plan recognition is done at two different levels. The first uses only what is believed to be shared knowledge between the speaker and hearer (i.e., the system). The second uses the above knowledge plus all other beliefs of the hearer. The goals inferred using shared knowledge were actually intended to be communicated by the speaker, and the hearer is obligated to respond to them. The goals inferred using all of the hearer's knowledge may be used in formulating a helpful response.

### Current Problem Areas in Speech-Act Theory

There are a large number of difficult technical problems that must be addressed before a fully adequate computational speech act theory can be formulated. Many of these involve basic issues of representation such as the representation of

belief, knowledge, and mutual belief, the representation of action, plans, and intentions, and the planning (qv) and plan-recognition processes themselves. Each of these stands as an independent research area in its own right and is not pursued further here. Instead, several issues of current concern more directly based in speech-act theory are examined.

The description of the treatment of indirect speech acts is based entirely in terms of recognized plans using shared knowledge. Curiously enough, a formulation of speech acts, except for the surface acts corresponding to the literal interpretation of the utterance, is not needed to specify the theory. In fact, introducing speech acts into the recognition process complicates the issue, for then problems arise such as how a single utterance can realize more than one speech act at a time and how a sequence of utterances can be said to realize a single speech act. If communication is defined just in terms of plans, the treatment of these situations is fairly straightforward. The intentions underlying a single utterance are characterized by a plan that may be described in several different ways in terms of speech acts. For example, the plan might satisfy the requirements of the speech act INFORM, as well as ANSWER, as well as WARNING. Each speech act describes a slightly different part of the plan that was recognized. Similarly, several utterances could incrementally build a plan that then could be described by a single speech act.

Thus, it may be that speech acts are useful only in language to summarize plans, clarify plans, or confirm plans, among other things. The speech acts themselves may not need to be present in the plan recognized. In an implementation, however, speech acts as plan summaries will probably remain as an important efficiency consideration; i.e., if one can recognize a speech-act description from an utterance, and entire chunk of a plan might be introduced at once. This becomes especially relevant when linguistic clues to the speech act being performed are considered. The word "please," e.g., signals a REQUEST no matter what the form of the sentence in which it is used.

The last example raises another important concern. What is the link between sentence structure and the speech acts that that sentence realizes? So far the mood of a sentence, and certain words such as "please," have been shown to influence the speech-act analysis. It is also true that certain phrases are much more likely to be used indirectly than others. For instance, consider the sentence "Can you pass the salt?" discussed above. How would things change if the utterance were "Are you able to pass the salt?" or even "Tell me whether you can pass the salt." Each one seems less likely to be treated indirectly. Current speech-act theories are at a loss to account for these differences in a principled way.

Finally, there is the issue of how speech-act theories relate to discourse models (see Discourse understanding). There is considerable work in segmenting discourse into fragments that reveal how sentences interrelate. Thus, one sentence might clarify the point in a previous sentence, introduce a new topic, or be an answer to a previous question. It seems that if these discourse models are to be useful computationally, some sort of recognition process is needed to relate the actual utterances to these discourse categories. The plan-recognition model for speech acts seems a good candidate in task-oriented domains, such as dialogues involving the performance of some task or in information retrieval. Can the speech-act models be extended in these domains to account for the behavior cap-



tured by these discourse models. And if so, how does that help in analyzing non-task-oriented conversations such as arguments and debate or casual conversations?

### Additional Readings

Cohen and Perrault (4) provide a good introduction to the plan-based theory of speech acts and its use in planning utterances. Appelt (5) extends this work to allow the realization of multiple speech acts by a single utterance. Allen and Perrault (6) and Allen (7) describe the use of plan recognition for analyzing speech acts as well as the model for indirect speech acts. Key pioneering work from philosophy includes references to Austin (1), Grice (3), and Searle (8,9).

### BIBLIOGRAPHY

1. J. L. Austin, *How to Do Things with Words*, Harvard University Press, Cambridge, MA, 1962.
2. R. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artif. Intell.* **2**, 189–208 (1971).
3. H. P. Grice, "Meaning," *Philos. Rev.* **66**, 377–388 (1957).
4. P. Cohen and R. Perrault, "Elements of a plan-based theory of speech acts," *Cog. Sci.* **3**(3), 177–212 (July–September 1979).
5. D. Appelt, *Planning Natural Language Utterances to Satisfy Multiple Goals*, Cambridge University Press, New York, 1985.
6. J. F. Allen and C. R. Perrault, "Analyzing intention in utterances," *Artif. Intell.* **15**(3), 143–178 (December 1980).
7. J. F. Allen, Recognizing Intentions from Natural Language Utterances, in M. Brady (ed.), *Computational Models of Discourse*, MIT Press, Cambridge, MA, 1983.
8. J. R. Searle, *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, New York, 1969.
9. J. R. Searle, Indirect speech acts, in P. Cole and J. Morgan (eds.), *Syntax and Semantics*, Vol. 3, *Speech Acts*, Academic Press, New York, pp. 59–82, 1975.

J. F. ALLEN  
University of Rochester

### SPEECH RECOGNITION

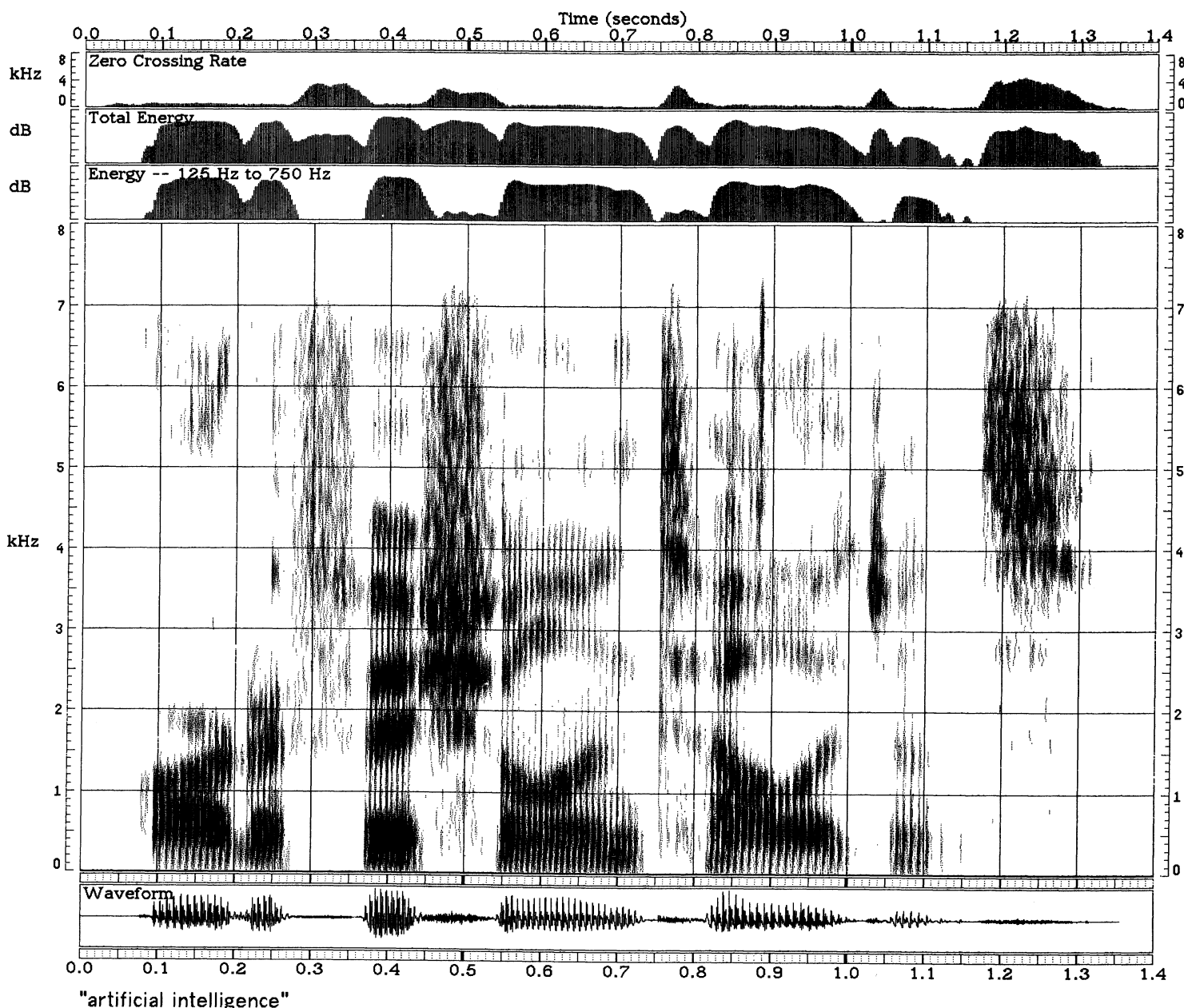
As computer systems have become more complex, it has been recognized that the use of these systems would be greatly expanded if human speech could be reliably utilized as an I/O medium. Speech synthesis (qv) is already being utilized for store- and forward-message services, as well as the capability of "reading" one's mail over the telephone from an arbitrary location. Speech recognition has had less utilization, and there is still an active search for appropriate applications that provide a useful functionality not readily available through other means. Such tasks include procedures where the hands are busy, as in package sorting, or where the hands may be soiled and inappropriate for writing, as in the fault diagnosis of greasy or rusty parts. Other applications include those where a wide variety of sensory modalities are being utilized under demanding real-time circumstances, as in an airplane cockpit. Speech recognition has also been utilized when the vocabulary is severely constrained to 10 digits, as in the conveyance of

credit-card numbers and other access codes for computer-based systems delivered over the telephone. Perhaps the most demanding task currently aspired to is that of dictation, where large vocabularies in excess of 20,000 words with a natural syntax are desired. This capability is not yet available commercially, but there are a number of companies striving to produce this functionality.

It is apparent from the discussion above that there are a wide variety of task parameters that must be considered in order to characterize a particular application of speech recognition. These include the particular vocabulary [including its phonemic (see Phonemes) content], the syntax utilized, discourse constraints (including the possible use of rigid scenarios), speaker selection (including age, sex, language of origin, and dialect), training demanded of the speakers, the particular kind of microphone (including considerations of directionality and microphone placement), the ambient environment in which the system is to work, and the way in which the system confirms input, indicates errors, and reacts in real time.

Currently available systems are based on the assumption that all of the necessary information needed for recognizing the spoken words is available in the speech signal itself. This implies that a pattern-recognition (qv) task based on some transformed property of the signal will be sufficient for recognition. Given the variety of actual speech signals, it has been impossible to effect the direct transformation from the signal to the linguistic symbols that are the desired output, even for small message sets. Instead, it has been necessary to first perform an information-reducing transformation so that the number of bits per second that is utilized in the pattern-matching techniques is reduced from as many as 50,000 bits/for high-quality digitally sampled speech to as few as 800 bits/for the spectral representation utilized in several high-performance systems. Not only does the spectral representation of speech obtained by computing the short-term Fourier transform of the windowed-input speech provide an information-reduced measure but it is also important because it focuses on important aspects of the speech that have been extensively studied in the field of experimental phonetics. An example of this representation is shown in Figure 1. The spectral representation is achieved by utilizing a "wide"-frequency-analyzing bandwidth, which averages over several harmonics of the speech signal, giving a smooth view of the spectral resonances and providing good resolution in time.

Although the spectral representation of speech is highly useful, it must be remembered that the signal under study varies greatly. This variation arises for many reasons, including variations in the size of the human vocal tract, the rate of speech employed by the talker, a variation in pronouncing patterns (even within a particular speaker), and the normal variability associated with the motion of the articulators (including the tongue, lips, jaw, and velum). It is this total array of variability that has provided the greatest challenge to the design of speech-recognition systems, and investigators have continually searched for transformations that will minimize this variability and place into evidence salient features that can be utilized in a pattern-recognition task. A variety of approaches have been utilized over the years to normalize the effect of varying vocal-tract size. These have generally involved some characterization of the space of trajectories of the vocal-tract articulators, including extreme points of articula-

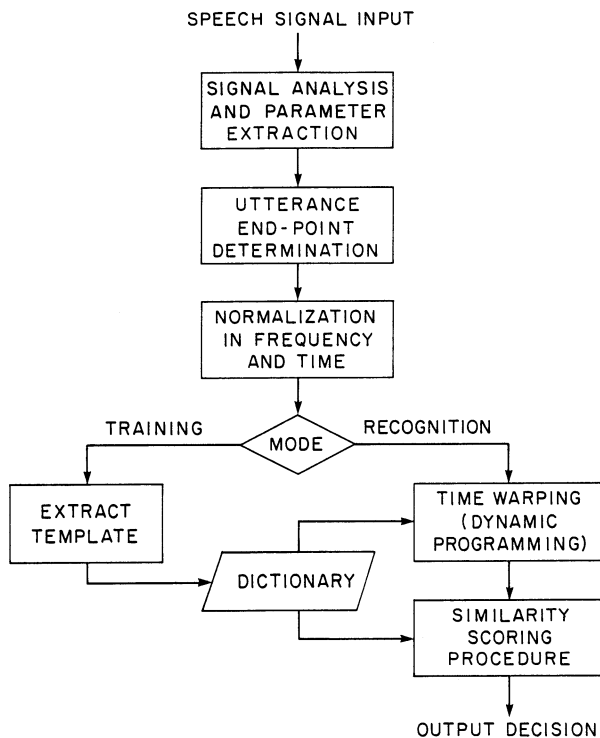


**Figure 1.** Wide-band spectrogram of the utterance "artificial intelligence." Time is plotted along the abscissa in seconds and frequency is plotted along the ordinate. Vertical striations show vocal-cord vibration, and resonances appear as dark bands. High-frequency sound indicates fricative noise.

tion as evidenced by the set of vowels utilized by the speaker. The most successful form of transformation utilized to minimize the effect of variability has been that of dynamic time warping, first introduced by Sakoe and Chiba (1) and refined in many ways to provide for efficient utilization. Essentially, the dynamic time-warping technique provides for local timing expansion and contraction in a way that minimizes the distance (as characterized by some chosen metric) between the transformed, warped-input spectral representation and that of a stored template for the particular speaker. The utilization of this technique has resulted in recognition accuracy improvements of as much as 20–30% over those techniques that do not utilize this approach. In fact, all commercially available systems exhibiting high accuracy over moderate-sized message

sets (e.g., NEC and TI) use some form of dynamic time warping.

The overall approach utilized by the pattern-matching (qv) techniques described above is shown in Figure 2. First, the signal is transformed into a spectral representation where a small informationally rich set of parameters can be determined. Next, end points are determined (this is not a trivial task), and normalizations are then applied in order to scale the parameters for varying-size vocal tracts as well as the characterization of overall speaking rate. During training sessions these transformed parameters are then used to form a template that is entered into a dictionary that characterizes the utterances within the message set for the particular speaker who is utilizing the system. Then, during actual recognition



**Figure 2.** Template-matching speech recognizer. The spectral template produced from the speech signal is normalized and compared with a set of templates derived from utterances of known identity. The input utterance is then assumed to be the same as that corresponding to the template with the highest similarity score.

tasks, the parameterized and normalized speech patterns are compared to these dictionary templates using dynamic time warping and some sort of similar scoring metric. It is well to recognize that the determination of these metrics is still an area of active study, and there is much room for improvement in this area.

It is clear that the spectral representation of speech is characterized by constraints of the human-vocal-tract apparatus and the way in which it is utilized by speakers. The most common way of modeling these effects is the source/filter model. The human vocal apparatus is modeled in terms of one or two sources exciting a set of coupled resonators that lead to peaks in the intensity of the sound in the neighborhood of particular frequencies called formants. During voiced sounds the vibration of the vocal cords, which creates puffs of air, is the excitatory source, and these short impulses excite the resonances of the vocal tract between the vocal cords and the lips; as the tongue, jaw, lips, teeth, and velum (nasal flap) move, the magnitude and location of these resonances change in a way that is characteristic of the particular sounds involved. On the other hand, the articulators may be positioned such that a very narrow constriction is placed in the vocal tract, giving rise to turbulent noise as the air is pushed through the vocal tract by the air pressure of the lungs. The sounds resulting from this form of excitation include the fricative consonants such as the /s/ in the word "son." Both forms of excitation can be combined, as in /z/. It is possible to build very accurate models of this sort and even to model directly the motion of the articulators in a physiologically realistic way. Utilization of these models has led to a great understanding of the way in which the speech signal is produced, but since the

articulators are hard to observe, there is still a great deal of ignorance remaining.

Although the nature of the vocal tract strongly constrains the output-speech signal, it is not the only constraint that is important to study, since much of the control of the vocal-tract muscles is voluntary and is determined by signals from the motor cortex of the brain. Though it cannot be systematically proved, it is probably true that all aspects of linguistic structure influence these control signals, and hence, the output-speech waveform, including the nature of the individual sound segments (vowels and consonants), syllable structure, morpheme structure (prefixes, roots, and suffixes), the lexicon, phrase- and clause-level syntax, and long-term discourse constraints. In the sequel there is some view of these constraining influences and the way in which they interact to produce the final speech signal. It is important to also recognize, however, that the nature of the human perceptual apparatus must also be modeled, providing additional important constraining influences. Only recently has there been appreciable study of the way in which the signal impinging on the eardrum is transformed through the middle-ear ossicles and excitation of nerve cells within the cochlea to form sequences of nerve spikes on the approximately 30,000 fibers of the auditory nerve. These nerve-firing patterns have only been studied for simple synthetic vowels, and a major new direction of research is being provided by the need to coordinate the study of speech production with the physiology of auditory perception in humans. Some models for the peripheral processing of the ear are now appearing (2), and it is reasonable to expect some improvement in speech-recognition performance due to the improved characterization of this influence.

Above the level of articulator control, the first level of language abstraction which is encountered is that of the individual phonetic segment, or at an abstract classification level, the phoneme (qv). Most natural languages have approximately 40 such elements, but they vary substantially over the numerous languages of the world. Thus, e.g., English vowels may be freely nasalized by talkers without affecting the intended phonemic percept, whereas in French the nasalization of vowels provides a phonemic contrast and hence affects the meaning of the utterance. This means that recognition strategies for English must be aware of the possibility of extensive nasal coarticulation that does not provide phonemic contrast, whereas in French nasal coarticulation into the domain of vowels may have substantial influence on phoneme perception and hence on the message meaning. Even though all human speakers have essentially the same vocal apparatus, it is put to astonishingly different usages, such as the utilization of tongue tip or velum trills and even the production of clicks, which exhibit phonemic contrasts in some African languages. It is clear that the nature of the articulator movements selected by a particular language to encode phonemic events, together with meaning contrasts provided with these phonemes, has a strong influence on the way in which the spectral representations of speech must be encoded at higher levels. These constraints are always actively utilized in practical systems. At the next level of linguistic structure phonemic segments are grouped together into consonant and vowel clusters and, in turn, into syllables. Furthermore, depending on the role of the phonemic segment within these syllables, their realization can change substantially. For this reason, what has sometimes been referred to as variation noise in the realization of these segments is in fact deterministically constrained by these structures and

can be exploited readily in powerful recognition strategies, although few investigators have utilized this constraint. Thus, e.g., an initial stop consonant in a syllable may be realized as substantially different than when it appears in final syllable position. Furthermore, consonant clusters provide very tight binding, and there are extremely strong sequential constraints imposed by a language on these clusters. For example, in English, if an initial consonant cluster consists of three phonemes, the first phoneme must be an /s/, the next phoneme must be an unvoiced stop consonant, and the third phoneme must be either /r/ or /l/. Examples of such occurrences would include "scrape" and "split." It is likely that native speakers are well aware of these constraints and can actively utilize them within the perceptual process.

It is evident from the above examples that although there are exceedingly strong constraints available to the listener at various points of an utterance, the strength of these constraints is not uniform over the duration of the utterance. This means that any modeling of the perceptual process must be active and provide for a focusing of the search, which seeks out those constraint domains as a function of time and can provide the greatest help in decoding the message. Another example of this need for focused search is found in the perception of stop consonants. Among the many cues for stop-consonant identity are the spectral nature of the noise burst at the release of the stop closure and the transition of the second formant resonance into the vowel following this closure. Many investigators have characterized these influences, but it has been shown that the constraining influence of these two characteristics on perception varies with the nature of the following vowel (3), and hence a powerful recognition strategy must have some knowledge of the rough position of a vowel succeeding a stop consonant before robust identification of the stop consonant itself can be made. These stop consonants also provide another illustration of the almost bewildering complexity of phonetic encoding utilized to mark linguistic distinctions. For example, in characterizing the distinction between the two words "rapid" and "rabid," as many as 16 different phonetic distinctions have been measured (4). The way in which these are utilized in production and exploited in perception has not been completely understood, and the combinatorics of the way in which these varying cues are integrated to form the intended percept is well beyond the current state of the art. In fact, it is the accurate characterization of this integrative process and the way in which it is utilized in perception that provides one of the major stumbling blocks to further progress in speech recognition.

Above the segment and syllable level, there are constraining influences due to the structure of morphemes (see Morphology), which are the minimal syntactic units of a language. These include prefixes, roots, and suffixes that must be combined in a prescribed way. It is thus possible to realize that there is a syntax both at the syllable level and at the morpheme level as well as the normally recognized syntax characterizing the way in which English words are combined into phrases and clauses. In fact, it is possible to represent these constraints as a nested sequence of context-free grammars (5), which naturally display the way in which phonetic segments vary as a consequence of this structure. In this way much of the "noisy" variability in speech segments has been related to this hierarchy of syntactic constraints.

Additional constraints on the nature of a lexical entry for a particular language are found at the word level. Several inves-

tigators have shown that a characterization of words in terms of five rough phonetic segment classes (6) can be expected to cut down the search space of possible words to a very small number, often providing unique identification. Furthermore, the effect of order of both letters and phonetic segments appears to be overemphasized, since in a study of French and English dictionaries it was found that over 90% of the words were uniquely defined simply by the unordered set of their composite letters, and for only 0.5% of these entries were there more than two alternatives (7). At the phoneme level it has been found that all words in a 20,000-word English dictionary are uniquely determined by their unordered phoneme pairs (8). These illustrations help to show that there is still a great deal of constraining influence at the lexical level that has not been exploited in contemporary speech-recognition systems. There is no question that continued research will reveal more of these constraints and that new control strategies for their utilization will be implemented.

Above the word level, syntax provides an additional constraining influence. Its affect on the sequential order of words is often characterized in systems by a branching factor that characterizes the number of possible words that can follow a particular previous word in an utterance. These constraints are also frequently represented in terms of a state diagram that indicates all possible transitions from word to word within a language subset utilized by a particular system. Syntax also has constraining influence on prosodic elements such as stress, as indicated by the way in which the stress of words such as "incline" and "survey" vary depending on the part of the speech they implement. In this way it is natural for each domain of constraint within a language to not only have a strong and consistent internal theory but also overlap with other constraint domains. Thus, although it is often possible to characterize the stress distribution of vowels within a word by considerations involving only the individual word, in the cases just shown the additional constraining influence of syntax must be considered. It is typical in natural language to find that none of these constraint domains allow a complete analysis within the confines of the domain itself but that significant overlaps such as those just shown occur between these domains. Once again, one can see the importance of an integrative process that actively and cohesively builds the intended percept from all of these constraining influences. Unfortunately, not only are people still ignorant of many of these constraining influences but they have very little understanding of the way in which such an integration should take place. Furthermore, above the level of syntax the constraint domains of semantics, pragmatics, and discourse are not at all well understood, and yet it is probably important to utilize these influences in particular task domains.

Despite the difficulty of characterizing and combining the constraints of the varying knowledge sources that have been described, a number of contemporary systems have compiled these influences into a single network (composite knowledge source) that represents all possible utterances. The HARPY (qv) system (9), developed at Carnegie-Mellon University, is an example of such a system, in which any allowed utterance can be described as a path through the compiled network. In this way constraints of syllable structure, word structure, and syntax are all bound into one fixed structure. Furthermore, the control structure used for search is an adaptation of dynamic programming techniques (which is also the basis of dynamic time warping) and is regarded as fixed. A more powerful

approach is provided by the use of hidden Markov models, which also utilize a single network where the probabilities associated with each branch can be accurately trained through extensive experience with a particular talker. Coded representations of spectral transformations of the input-speech waveform are utilized in this approach to find the most probable path through the network, and very good results have been recently obtained (10). It is important to emphasize the utilization of such a formally structured approach, which provides for the automatic determination of symbolic classes through clustering, and also maximizes performance through sophisticated training procedures. Without such automatic clustering and parameterization, it is probably impossible to achieve high performance in the complex, contemporary systems studied today.

In another approach, however, the knowledge sources and their associated processes are loosely coupled and flexibly utilized by a sophisticated control structure. This approach is typified by the HEARSAY-II (qv) system (11) developed at Carnegie-Mellon University and the HWIM (hear what I mean) system (12) constructed at Bolt, Beranek and Newman. In these systems a complex data structure that contains all of the information discovered about an utterance is maintained and is visible to a set of processes, each of which is attempting to discover the structure associated with a particular constraint domain. As indicated above, each of these constraint domains has a substantial internal theory, but a complete analysis is not available entirely within any one of these domains. For this reason, the common unifying data structure must provide for the interaction between these processes, and it is clear that a complex control structure must be provided for this integration. Although these frameworks are very general and seem well motivated by the existence of multiple knowledge sources and their impact on speech perception, they also provide a very large number of degrees of freedom that must be understood and effectively controlled for proper system performance. At this time, training and tuning procedures for these systems are largely heuristic, and effective means must be derived for the characterization of these complex control strategies. By contrast, the techniques based on Markov chains utilize a strong mathematical discipline with well-understood training procedures. The challenge presented to all of these systems is to provide the requisite capability of focused search over a number of interacting and integrating constraint domains in the context of a sufficiently rich but constrained formalism that permits both adequate parameterization and comprehensive training. Those systems providing explicitly interacting constraint domains have focused largely on knowledge representation and are relatively weak on training and control, whereas those systems that are embedded in a rigid mathematical framework that provides excellent techniques for parametric determination and training optimization have been lacking in the utilization of complex data structures needed to characterize high-level constraints such as syntax. In the future some convergence of these techniques can be expected as investigators learn to exploit the best features of each of these approaches.

In conclusion, the influence of implementation technology on these systems should be mentioned. It is probably true that comprehensive solutions to the speech-recognition problem will require computing rates of at least 100 MIPS (million instructions per second), as well as approximately 100–1000 Mbytes of real memory. These are large numbers by today's

standards, but there is a vast ongoing improvement in technology, and by the start of the next decade, even single-chip microprocessors will provide performance levels of 50 MIPS, and 64-Mbit memory chips will also be available. For this reason, the underlying integrated circuit technology itself does not seem to be a major problem for speech-recognition systems. Instead, it is the overall architecture of these systems, including the way in which constraints are represented and exploited, that must be determined. Large experiments must be carried out and new ways found to provide for the needed interaction of constraining influence. In many ways speech recognition provides an archetype example of the emerging class of highly complex systems that must utilize the best techniques of modern computer science and AI.

## BIBLIOGRAPHY

1. H. Sakoe and S. Chiba, "A Dynamic Programming Approach to Continuous Speech Recognition," *Proceedings of the International Congress of Acoustics*, Budapest, Hungary, 1971, Paper 20C-13. [A more recent view of these techniques is C. S. Myers and L. R. Rabiner, A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition, *IEEE Trans. Acoust. Speech Sig. Process.* **ASSP-29**(2), 284–297 (April 1981).]
2. S. Seneff, "Pitch and Spectral Analysis of Speech Based on an Auditory Synchrony Model," Ph.D. Thesis, MIT Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, 1985.
3. E. Fischer-Jorgensen, "Tape-Cutting Experiments with Danish Stop Consonants in Initial Position," Annual Report, Institute of Phonetics, University of Copenhagen, 1972.
4. L. Lisker, Rapid vs. Rigid: A Catalogue of Acoustic Features That May Cue the Distinction, Haskins Laboratories Status Report, SR-54, New Haven, CT, 1978.
5. K. W. Church, "Phrase Structure Parsing: A Method for Taking Advantage of Allophonic Constraints," Technical Report 296, MIT Laboratory for Computer Science, Cambridge, MA, 1983.
6. D. W. Shipman and V. W. Zue, "Properties of Large Lexicons: Implications for Advanced Isolated Word Recognition Systems," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Paris, France, 1982, pp. 546–549, 1982.
7. G. Deloche, F. Debili, and E. Andreewsky, "Order information redundancy of verbal codes in French and English: Neurolinguistic implications," *J. Verb. Learn. Verb. Behav.* **19**, 525–530 (1980).
8. S. M. Marcus, Associative Models and the Time Course of Speech, in M. Schroeder (ed.), *Speech and Speaker Recognition*, Karger, Basel, pp. 36–52, 1985.
9. B. T. Lowerre, "The HARP Y Speech Recognition System," Technical Report, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 1976.
10. F. Jelinek, "The development of an experimental discrete dictation recognizer," *Proc. IEEE*, (1987).
11. V. R. Lesser, R. D. Fennell, L. D. Erman, and D. R. Reddy, "Organization of the HEARSAY II speech understanding system," *IEEE Trans. Acoust. Speech Sig. Process.* **ASSP-23**, 11–24 (1975).
12. W. A. Woods, "Language Processing for Speech Understanding," in F. Fallside and W. A. Woods (eds.), *Computer Speech Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

## General References

- W. A. Lea, *Trends in Speech Recognition*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
- D. R. Reddy (ed.), *Speech Recognition*, Academic, New York, 1975.

F. Fallside and W. A. Woods, *Computer Speech Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

J. L. Flanagan, *Speech Analysis, Synthesis, and Perception*, Springer, New York, 1972.

J. ALLEN  
MIT

**SPEECH RECOGNITION AND GENERATION.** See Natural-language generation; Natural-language understanding.

## SPEECH SYNTHESIS

Over the years there has been an increasing need for speech generated from computers. In part, this has been due to the intrinsic nature of text, speech, and computing. Certainly, speech is the fundamental language representation, present in all cultures (no matter whether literate), so if there is to be any communication means between the computer and its human users, speech provides the most broadly useful modality, except for the needs of the deaf. Although text (seen as a string of conventional symbols) is often considered to be more durable than speech and more reliably preserved, this is in many ways a manifestation of the relatively early progress in printing technology, as opposed to the technology available for storing and manipulating speech. Furthermore, text-based interaction with computers requires typing (and often reading) skills that many potential users do not possess. So, if the increasingly ubiquitous computer is to be useful to the largest possible segment of society, interaction with it via natural language (and in particular, via speech) is necessary. In other words, there is a clear trend over the last 25 years for the computer to bend increasingly to the needs of the user. This accommodation must continue if computers are to serve society at large.

The current search for expressive, easy-to-use programming languages that are not prone to error can be expected to lead, in part, to natural-language interaction as the means best suited to human users, with speech as the most desirable mode of expression (see Natural-language interfaces).

### Constraints on Speech Synthesis

Given that speech communication with computers is both needed and desirable, we can ask what the nature of these techniques is, and how they are realized. In order to get an overview of these procedures, it is useful to consider them as the result of four different constraints that determine a design space for all possible speech-output schemes. Each technique can then be seen as the result of decisions related to the impact of each of the four constraint areas.

**Task.** The desired application task determines the nature of the speech capability that must be provided. When only a small number of utterances is required (and these do not have to be varied on-line), recorded speech can then be used. On the other hand, if the task is to stimulate the human cognitive process of reading aloud, an entirely different range of techniques is needed.

**Human Vocal Apparatus.** All speech-synthesis systems must produce a speech waveform as output, but this is not an arbitrary

signal. Much effort has gone into the efficient and insightful representation of the speech signal resulting from a signal source in the vocal tract that excites the vocal-tract "system function," which acts as a time-varying filter to produce the speech waveform (1-3). The articulators of the human vocal tract also constrain the speed with which signal changes can be made. They are responsible for much of the coarticulatory smoothing (or encoding) that makes the relationship between the underlying discrete phonetic transcription and the speech waveform so difficult to characterize.

**Language Structure.** Just as the speech waveform is not an arbitrary signal, the myriad possible speech gestures that could be related to a linguistic message are constrained by the nature of the particular language structure involved. It has been consistently found that the units and structures used by linguists to describe and explain language do, in fact, provide the appropriate base in terms of how the speech waveform can be characterized and constructed. Thus, basic phonological laws, stress rules, morphological and syntactic structures, and phonotactic constraints all play a part in determining the speech output.

**Technology.** The ability to successfully model and implement speech-output devices is strongly conditioned by current (and previous) technology. Speech science has profited greatly from a variety of technologies including X-rays, motion pictures, modern filter and spectral estimation theory, and most importantly, the modern digital computer. The advent of increasingly capable integrated circuit technology has made it possible to build powerful, compact, low-cost, and real-time devices. This fact, combined with the substantial knowledge of algorithms needed to generate speech, has propelled the field of speech output from computers into the real world of practical and commercial systems suitable for a wide variety of applications.

### Synthesis Techniques

Keeping these constraints in mind, consider the various approaches to speech output from computers. Many techniques have been developed, but they can be naturally grouped in an insightful way. The purpose is to create a context in which the various techniques for speech synthesis can be considered (ranging from simple waveform coding to unrestricted systems for the conversion of English text to speech). This comparison will allow the difference between the various approaches to be highlighted, as well as comparison of system cost and performance.

**Waveform Coding.** The simplest strategy for speech synthesis is to record the desired speech (either in digital or analog format) and then play it back on demand according to the needs of the application. Depending on the technology used, this approach may introduce access time delays and will be limited in capacity by the recording medium available. But the speech will generally be of high quality. No knowledge of the human vocal apparatus or language structure is needed. These systems are a straightforward match of the task requirements to the available storage technology. Since memory size is the major limitation of these schemes, efforts have been made to decrease the number of bits per time for digital stor-



age. A variety of techniques has been used that includes simple data modulation, adaptive delta modulation, adaptive differential pulse code modulation, and adaptive predictive coding (4). These techniques can decrease the required bit rate from over 50 kbits/s (when no coding is used) to under 10 kbits/s, while good speech quality is retained. In general, coding and decoding complexity increases as the number of bps decreases. There is still much active research in this area.

Integrated circuits for coding and decoding in these applications are readily available in contemporary technology, and they are easy to interface with practical systems for recording and playback. When the message vocabulary is small and fixed, these systems are attractive. But if messages must be concatenated, it is extremely difficult to produce good-quality speech because aspects of the speech waveform have been "bound" to values that may not be suitable in a larger context not anticipated at the time of the original recordings. In all speech-synthesis schemes there is a general trade-off between speech quality and flexibility. The increase in flexibility inevitably also leads to greater computational complexity.

**Parametric Representation.** In order to further reduce storage requirements (and to provide the needed flexibility for concatenation of messages) several schemes have been developed that abstract from the waveform itself to a parametric representation in terms of a model for speech production. These parameters may characterize salient information in either the time or frequency domain. For example, the speech waveform can be formed by summing up waveforms at several harmonics of the intended pitch weighted by the spectral prominence at that frequency. Alternatively, a set of resonances can be excited by noise or glottal waveforms, or the vocal-tract shape can be simulated along with appropriate acoustic excitation. When compared to waveform coding, more computation is required at the time of playback to recreate the speech waveform, but the storage requirements per message are reduced. More important, the parametric representation provides an abstraction on the speech waveform to a level of representation where the attributes that contribute to speech quality (e.g., formant resonance frequencies and bandwidths, pitch, and excitation amplitudes) can be successfully manipulated. This allows elementary messages to be concatenated in a way that provides for smooth transitions at the boundaries. It also allows for changes in attributes such as pitch throughout the individual message units so that substantial changes in timing and intonation can be readily made. The most popular parametric representations currently in use are based on formant resonances or linear predictive coding (LPC), although vocal-tract articulatory models have been used in experimental systems (5). Integrated circuit chips are available to provide for the storage of speech in terms of these parameters, together with the control and resynthesis of the speech from these parameters in real time. These are inexpensive and widely used in many audio-announcement schemes. Message units of widely varying sizes are employed, which include paragraphs, sentences, phrases, words, syllables, demisyllables, and diphones. As the size of the message unit is reduced, fewer basic messages are needed for a large message set. But more computation is required, and the difficulties in correctly representing the coarticulation across message boundaries increase. Clearly, these schemes attempt to preserve as much of

the quality of natural speech as possible and permit the flexible construction of a large set of messages using elements that require little storage. In addition, individual speaker identity is preserved. With the current knowledge level of digital signal-processing techniques and the accompanying technology, these schemes have become very important in practical applications. It is important to remember that parametric representation systems seek to match the task with the available processing and memory technology by using the knowledge of human-speech-production models. But little (if any) use is made of the linguistic structure of the language.

**Synthesis-by-Rule.** When messages are concatenated by using parametric representations, there is a trade-off between speech quality and the need to vary the parameters in order to adapt the message to varying environments. Researchers have found that many pronunciations of a message unit (e.g., a syllable) may be needed to achieve good-quality speech, leading to large vocabularies of units with little insight as to the contextual situations that specify a particular choice. For this reason, the synthesis process has been abstracted beyond the level of parametric representation to a set of rules that seek to compute the needed parameters for the speech-production model from an input phonetic description (6). This input representation contains very little information in itself. Usually the names of the phonetic segments (e.g., the vowels and consonants), together with stress marks and indicators for pitch and timing, are provided. But it is also possible to compute these last two correlates from the identity of the individual segments, the syntactic structure, and stress marks, plus semantic information if it is available. In this way synthesis-by-rule techniques can utilize a very low bit rate (less than 100 bits/s) message description as input. Substantial computation must be used to compute the model parameters to produce the speech waveform. There is complete freedom to specify the model parameters, but there is also the need to control these parameters correctly. It must be emphasized that the available rules for this purpose are imperfect. The resulting speech is not as good as recorded human speech, but recent tests have shown that high intelligibility and comprehensibility can be obtained. When sentence- and paragraph-level messages must be synthesized, the rule system provides the necessary degrees of freedom to produce smooth, flowing, good-quality speech that is naturally inflected. From a computational viewpoint, it is interesting to consider that synthesis-by-rule systems delay the binding of the speech parameter set and waveform to the input message by using very deep language abstractions and hence provide maximum flexibility. For this reason, they are well suited to the needs of converting unrestricted text to speech. On the other hand, the designers of these systems must discover the relationship between the underlying linguistic specification of the message and the resulting speech signal. This topic has been studied in speech science and linguistics for several decades. Thus, synthesis by rule both benefits from and contributes to the general knowledge of speech and linguistics. The steady improvement in speech synthesis-by-rule quality reflects this joint progress. Although it is believed that current synthetic speech quality is acceptable for many applications, it can be expected to continually improve with the increased knowledge of how linguistic structure is related to speech articulation.

### Text-to-Speech Conversion

The synthesis-by-rule techniques previously described require a detailed phonetic transcription for input. Although this input requires very little memory for message storage, expert knowledge is required to derive the appropriate parameters. In many applications a frequent requirement is to convert text to speech (7-9). When it is desired to convert unrestricted English text to speech, the flexibility of synthesis by rule is needed, so some means must be afforded to convert the input text to the phonetic transcription needed by rule techniques. First, the text must be analyzed to obtain the phonetic transcription, which is then subjected to a synthesis procedure to yield the output speech waveform. The analysis of the text is often linguistic in nature and involves a determination of the underlying phonemic, syllabic, morphemic, and syntactic form of the message plus whatever semantic and pragmatic information can be gleaned. Text-to-speech conversion can be seen as a collection of techniques that requires a successful integration of the task constraints with other constraints provided by the nature of the human vocal apparatus, the linguistic structure of the language, and the implementation technology. It is the most complex form of speech-synthesis system, but also the most fundamental in design and most useful in application since it seeks to mirror the human cognitive capability for reading aloud. Other cognitive models attempt to synthesize speech directly from "concept" for those applications where the underlying linguistic structure is readily available (10). These schemes (presumably) have the advantage of more detailed syntactic and semantic structure than can be obtained from text, and hence they are of great interest for high-quality synthesis. But the pervading presence of text in one's culture makes the text-to-speech capability of great practical importance. It is worth emphasizing that both text and speech are surface manifestations of underlying linguistic form, and the text-to-speech conversion consists of discovering that underlying form and then utilizing it to form the output speech.

### MITalk Text-to-Speech System

In the previous sections several constraints that influence the design of speech-synthesis systems have been described, and text-to-speech systems are by far the most complex of these conversion methodologies. By focusing on a single text-to-speech system, it is possible to illustrate these many constraining influences and to provide the reader with an insight into the strengths and limitations of commercial offerings. Starting in the late 1960s the MITalk system was developed at MIT (11-13). Its goal was to convert unrestricted English text to speech. The original motivation for this work stemmed from the desire to build a reading machine for the blind, but the MITalk system is completely general and readily applicable to any situation where there is a need to convert text to speech by computational means. There have been several approaches to this task, but the MITalk system provides the most comprehensive set of techniques and is distinguished by its innovative use of morphological analyses, letter-to-sound rules, lexical-stress rules, and both prosodic and phonemic synthesis strategies. In the sequel a view is given on the various computational modules of this system, always keeping in mind the requirements that the system provides intelligible and readily comprehensible speech from unrestricted English text. In fact,

it has always been the goal of the MITalk system to model human performance in reading aloud. Although this is a staggeringly complex cognitive task, the MITalk system (apart from its practical applications) provides important insights into one's understanding of this cognitive task.

### Analysis of Text

**Conversion of Symbols to Standard Form.** Any examination of the wide variety of printed text will show that there are symbols and abbreviations that do not constitute "normally spelled words." It is not desirable to subject the succeeding linguistic-analysis algorithms to such symbols, so a conversion must be utilized in order to convert these symbols to normal (often expanded) forms. Thus, symbols such as "%" and "&" and abbreviations such as "Mr." and "Nov." must be converted to a normal form. There is a surprisingly large number of situations that must be converted, and careful routines have been established for the derivation of the spoken form of alphanumerics, numbers, dates, and money amounts. There are occasional ambiguities, such as the use of the hyphen symbol at the end of a line. It seems clear that human readers often use a great deal of contextual and pragmatic knowledge to determine appropriate pronunciations that are not available to the text-conversion algorithms. Nevertheless, such normalization techniques have been satisfactorily derived, and although they are not fundamental or deep in any linguistic sense, they are effective for purposes of the overall text-to-speech conversion.

**Morphological Analysis.** Consider the problem of converting unrestricted, but normalized, English text to a synthetic speech waveform. In the input text, word boundaries are readily detectable so that one might allow for storing the pronunciation of all English words. This number is large, but bounded, so the size of the dictionary would be large. But there are several attractive features of this approach. First, some form of dictionary must be used to provide pronunciations for exceptions to other rules used to derive pronunciations. These arise, in part, from foreign words that have retained a pronunciation of their language of origin (e.g., *parfait* and *tortilla*). Furthermore, all mechanisms derived thus far for the conversion of letter strings to phonetic segment labels provide some errors. It seems inherent in natural languages that no formal means derived for the representation of their structure has covered all observed forms without error. An interesting class of exceptional pronunciation arises for words that are frequently used. For example, initial /th/ is pronounced as a voiceless fricative in many words (e.g., *thin*, *thesis*, and *thimble*). But for very frequent words, such as the short function words (e.g., *the*, *this*, *there*, *these*, *those*, etc.) it is pronounced in a voiced manner. Similarly, /f/ is always pronounced as an unvoiced fricative, except for the single case "of." In words such as "shave" and "behave" the final silent /e/ has the effect of lengthening the preceding vowel, but in the frequent word "have," this is not the case. Finally, the final /s/ in "atlas" and "canvas" is unvoiced, but for the function words (e.g., *is*, *was*, and *has*), it is voiced. It appears that these high-frequency words should be placed in an exceptions dictionary if a set of rules is to be used for converting letter strings to phonetic segment labels. From the above discussion it is clear that some form of exceptions dictionary is necessary. Given that all systems should provide such a lexicon, there are two choices in handling nonexcep-

tional words. On one extreme, system designers could attempt to provide a "complete" word dictionary. Unfortunately, although the number of words is bounded, new words are constantly invented by productive processes of compounding (e.g., earthrise and cranapple) and by the filling out of "accidental gaps" (in the phonological sense) as in brillig. In addition, a comprehensive word lexicon would be required to store all regularly inflected forms, which places a large burden on the storage required. So, a "complete" word lexicon is not satisfactory. This fact has led investigators to consider the other extreme, namely the provision of a set of letter-to-sound rules that would convert input letter strings to phonetic segment labels through some sort of scanning and transformation process. Such rule sets have been constructed (MITalk has an extensive set), and they are very productive. But difficulties have remained. It is difficult to provide a high degree of accuracy from these rule sets, leading to increases in the size of the "exceptions" dictionary. These problems arise, in part, due to the fact that there is internal structure in words that must be recognized in order to derive the correct pronunciation.

It has been found that the minimum syntactic unit of language, the morpheme (see Morphology), has an important role to play in the determination of pronunciations. When morphemes are represented by letter-string segments called "morphs," they can be effectively used as the basis for determining word pronunciation. MITalk uses a morph lexicon that can be viewed as a bridge between the two extreme approaches mentioned above. Together with an effective analysis procedure, this lexicon provides for accurate pronunciations, including exceptions, and also provides a natural role for letter-to-sound rules that must be present in order to convert unrestricted English text to speech.

Many English words can be recognized as being constructed of a succession of morphs such as prefixes, roots, and suffixes. Thus, the word "snowplows" has two roots and an inflectional suffix; the word "relearn" has a prefix and a root; and "antidish-establishmentarianism" has no fewer than seven recognizable morphs. These morphs are the atomic constituents of words, and they are relatively stable in a language. They are often the ingredients of newly coined compound words, but new morphs are rarely formed. For this reason, they are good candidates for lexical entries, provided that a means can be found to analyze words into their constituent morphs. An effective morph lexicon can have less than 10,000 entries so that reasonable storage efficiency is provided, particularly in contemporary integrated circuit technology. It is also important to note that with a morph lexicon and the associated analysis procedures there is no need to store all of the regularly inflected forms, as is the case with the whole-word lexicon.

Since morphs are the basic constituents of words, it is important to illustrate their utility in determining pronunciation. When morphs are joined, they often change pronunciation depending on the nature of the morphs involved. Thus, when the plural form of the singular nouns "dog" and "cat" is realized, the final /s/ is voiced in "dogs" but unvoiced in "cats." This is a form of morphophonemic rule concerned with the realization of the plural morpheme in various environments. In order to use these rules, it is necessary to recognize the constituent morphemes of a word, so it is apparent that there is an important class of pronunciation effects that is facilitated through the detection of morphs and their boundaries. MITalk provides a comprehensive implementation of the morphophonemic rules of English.

Another advantage of morphemic analysis is the provision of an appropriate base for the proper utilization of letter-to-sound rules. Most sets of letter-to-sound rules treat each word as an unstructured sequence of letters and use a scanning window to find consonant and vowel letter clusters that can be readily converted to phonetic segment labels. The two letters "t" and "h" often occur as a consonant cluster, but in the word "hothouse," the /th/ cluster is broken up by a morph boundary. Similarly, the vowel digraph /ea/ presents many pronunciation difficulties for a letter-to-sound algorithm, but in the word "changeable" it is clearly broken up. In the MITalk system morph analysis is always attempted before letter-to-sound rules are used. Care is taken to ensure that letter-to-sound rules are not applied across morph boundaries. Consequently, not only does morph use lead to an efficient and productive lexicon, it also naturally provides for important pronunciation effects due to morph structure. It also sets an appropriate basis for the formulation of a well-motivated set of letter-to-sound rules, devoid of ad hoc exceptions.

The underlying abstract morphemes of a word are not always readily apparent as segments of a word. The sequence tall, taller, and tallest can be readily analyzed into identifiable segments, but many morphemes such as PLURAL are not easily detected, as in "mouse" and "mice" or "fish" and "fish." When such segmentation is not possible, all of the various forms must be placed in the lexicon. But these so-called "strong" derivations are few and do not cause excessive swelling of the lexicon's size.

An important part of the MITalk system is the provision of the morph lexicon and the corresponding analysis algorithm. A great deal of optimization has gone into this algorithm, the detailed description of which is beyond the scope of this entry (13). Between 95 and 98% of the word tokens of normally occurring English text are satisfactorily analyzed by morph-analysis procedures to provide the phonetic pronunciation of the word and an estimation of its parts of speech. These techniques work exceedingly well and are responsible for much of the high quality of the phonetic segment labels utilized in the MITalk system.

**Letter-to-Sound Rules and Lexical Stress.** In the MITalk system each normalized input text word is first subjected to morphemic analysis. It may be that the entire word is in the morph lexicon, such as the word "snow." On the other hand, the word may be analyzed into a concatenated sequence of morphs, each of which is found in the morph lexicon. The average number of morphs per word in English is approximately two. In the event that neither the entire word can be found in the lexicon nor can it be analyzed into a succession of morphs, a set of letter-to-sound rules is utilized to determine the sequence of phonetic segment labels that corresponds to the text form of the word. It is important to emphasize that these techniques are never used unless the preceding morphemic analysis fails. The conversion of a letter string to a phoneme string by using the letter-to-sound rules proceeds in three stages. In the first stage prefixes and suffixes are detected and removed, since letter-to-sound rules do not apply across morph boundaries. This affix-stripping capability is not as strong as that found within the morphemic analysis, but it is satisfactory for this purpose. It is assumed that after the prefixes and suffixes are removed, the remaining, central portion of the word consists of a single morph. It can then be subjected to letter-to-sound rules. The second phase of the

rules converts the consonants of this central portion to phonetic segment labels, starting with the longest recognizable consonant clusters and continuing until all single-letter consonants are converted. Finally, the remaining vowels in the central portion are converted by using both letter and phonetic segment symbol contexts. The vowels are converted last because they are the most difficult and context-dependent in terms of pronunciation determination. For example, the vowel digraph /ea/ has as many as 14 different pronunciations as seen in the words "reach," "tear," "steak," and "leather."

A great deal of research effort has been spent to determine these letter-to-sound rules, but in the MITalk system they are applied only to morph units due to the utilization of the preceding morphemic analysis. Furthermore, in MITalk the letter-to-sound rules are coupled to an extensive set of rules for the placement of lexical stress. As recently as 25 years ago, competent linguists did not believe there was any systematic way to distribute stress over English words (see Computational linguistics). Now, however, as a result of phonology research and further extensions in MITalk, an extensive set of rules for stress placement has been developed. There is some stress variation due to the syntactic role of words, such as the adjectival use of the word "invalid" as opposed to its nominal use. These words are few, but they are important to a high-quality system. In addition, some suffixes automatically take on the main stress of a word, such as the suffix in the word "engineer." Beyond these relatively simple cases, there are many highly complicated situations that must be treated by a set of cyclic rules, the nature of which is beyond the scope of this entry. Nevertheless, these have been extensively implemented within the MITalk system and are indispensable to the resultant speech quality. For example, the differences in pronunciation between the words "systematic" and "systematize" are readily computable, as is the shift in stress between the word "human" and "humanity."

Several sets of letter-to-sound rules have been developed for MITalk, the most comprehensive of which includes 600 rules. As seen earlier, these rules are rarely used, but they must be present in order to correctly convert unrestricted English text to speech. In MITalk it is expected that all strong and irregular forms will be converted by using the morph lexicon. The letter-to-sound rules are used for less frequently occurring, newly invented, and occasionally misspelled words. For example, in MITalk, the misspelled word "recieved" is correctly pronounced due to the use of the letter-to-sound rules.

**Parsing.** Every comprehensive scheme for the conversion of unrestricted text to speech must include techniques for syntactic analysis (see Parsing). The correct syntactic role of each word must be determined since it often affects the corresponding phonetic segment as well as its stress. Furthermore, the use of parsing is absolutely essential in order to determine the correct timing and pitch contour. These so-called prosodic correlates are essential in generating speech that will sound interesting and lively and that can be listened to for long periods of time. Unfortunately, it is beyond the state of the art in computational linguistics to obtain a complete clause-level parse for unrestricted English text. In addition, many sentences are ambiguous at the syntactic level, and yet only one pronunciation can be produced by the overall text-to-speech system. Nevertheless, it is possible to perform phrase-level parsing, which can produce much of the needed structure for

purposes of speech synthesis but will (in some situations) be in error due to the lack of the complete clause-level analysis. In MITalk such a phrase-level parser is used. Of course, there are many multiply ambiguous sentences, such as "he saw the man in the park with a telescope," for which phrase-level analysis is quite satisfactory. The performance of these phrase-level analysis routines would improve considerably given a more accurate determination of the individual word parts of speech.

In English the strongest syntagmatic constraints (i.e., constraints on the left-to-right sequence of words) occur within auxiliary-verb phrases and determiner sequences in noun phrases. The MITalk system takes advantage of this fact and provides highly accurate augmented-transition-network grammars (see Grammar, augmented-transition-network) for these environments. Thus, MITalk seeks to maximize correctness where it can be readily achieved at the phrase level, although it does not seek to build more elaborate clause-level structures. This approach has been satisfactory, but an efficient clause-level parser (which fails gracefully when an incomplete or ambiguous parse is obtained) would undoubtedly enhance the system's performance. After completion of the parsing activity, the system is able to mark the functioning parts of speech for all words and indicate all syntactic breaks in the sentence as a basis for further refinements in pronunciation, timing, and pitch.

**Stress Modification and Phonological Adjustments.** The last part of MITalk's analysis phase consists of several minor adjustments to the computed phonetic segment labels corresponding to a variety of contextual environments. A simple example is the correct determination of the pronunciation of the word "the," which depends on the phonetic segment at the beginning of the following word. In addition, several heuristic techniques are used to ensure that the overall sentence-level stress contour is appropriately related to the contours of the individual words and that pauses are inserted at natural places so that the artificial speech does not appear to be running out of breath. These modifications can be regarded as smoothings and have rarely been incorporated into a well-founded theory. But there is a substantial body of knowledge about their occurrence, and they are important to the overall quality of output speech. Following the utilization of these rules, the overall analysis phase is seen as having converted the input set of text to a string of phonetic segment labels complete with stress marks, syntactic breaks, and punctuation indicators. This new representation is the base on which the synthesis activities are formed. It may be seen as the common underlying representation that links the text and speech surface representations of language.

### Synthesis

Thus far, the procedures utilized to obtain the underlying linguistic representation common to both text and speech have been described. The output-speech waveform remains to be synthesized from this underlying description. In the MITalk system it is important to realize that no stored waveforms are utilized even in parametric representation. Instead, rules that control the parameters have been devised in such a way that the desired output-speech waveform will be realized. This is a computationally intensive task, but it avoids the need to store parametric representations of many morphs or words.

**Prosodic Framework.** The first step in building the output-speech waveform is the generation of a framework for timing and fundamental frequency (the major correlate of intonation) on which the detailed articulations of the individual phonetic elements can be built. Failure to realize an appropriate prosodic framework will lead to highly stilted and unnatural speech that cannot be listened to for long periods. In many ways, the stress distribution, which has previously been calculated in the analysis phase, is responsible for the main features of the timing contour as well as the pitch contour. The casual observer often feels that intensity is the main correlate of stress, but instead, duration and perturbations of the pitch contour are the main cues used by native-English speakers. Intensity is far less important. Keeping this in mind, the intrinsic nature of the individual phonetic segments together with stress marks, syntactic markers, and punctuation are used to generate an overall timing pattern. The consonants do not change greatly over time as a function of stress, but the vowels are very plastic and can be easily compressed and stretched according to the needs of the overall stress distribution. There is also a tendency for speakers to slow down as they near major breaks in a sentence and to compress intervals of relatively unstressed segments. In the MITalk system several rules that reflect these tendencies have been implemented, resulting in a natural timing contour. In addition, the fundamental frequency (or pitch) contour must be generated on the timing framework. For declarative sentences there is a general tendency for the pitch to rise quickly on the first stressed syllable and then to diminish slowly along a "declination line" toward the last stressed syllable, where the contour falls rapidly to the end of the sentence. Questions result in different contours, as do imperatives. In addition to the overall contour of each sentence, there are many pitch perturbations concerned with the segmental nature of the speech and the local effects of stress. There is also a tendency for new information to receive more stress than old information and for words that express negation and doubt (e.g., might) to receive increased values of fundamental frequency. MITalk includes a comprehensive algorithm for realizing these features, although it is recognized that pitch is also used to mark semantic contrast and emotive emphasis, which cannot be derived directly from text. The importance of obtaining an appropriate prosodic contour must be strongly emphasized since an inappropriate form will mislead the listener and cause great perceptual difficulty. There is still a need for vastly increased understanding of the prosodic framework, but sufficient knowledge is available in contemporary practice to provide for acceptable synthetic speech in many practical applications.

**Synthesis of Phonetic Segments.** Once the prosodic framework is complete, the parameters corresponding to the vocal-tract model in use must be generated. This will permit the speech waveform to be generated. Typically, there are 25 parameters that vary with time that are updated on 5- or 10-ms intervals. Given the names of the phonetic segments and the prosodic framework previously described, a series of target values for these parameters is generated for each segment. Then, a set of smoothing rules is used to generate values for the parameters at each of the sampling intervals. In current practice, as many as 100 contextual rules are utilized to properly describe the trajectories of these parameters. There is still the need for increased understanding that can only be obtained through the systematic study of very large phonetic

databases. Once the sample values of the parameters are obtained, they must be transmitted at the sampling interval to the appropriate model of the vocal tract (usually a formant model or an LPC model), which can then be exercised to produce the speech waveform. At this point there is much signal-processing computation that must be performed, but it can readily be accommodated on a single digital signal-processing integrated circuit chip. The output discrete samples, generally produced at a 10-kHz rate, are then smoothed and passed through an audio amplifier to produce the final output speech.

### Evaluation of Synthetic Speech

Given the commercial interest in synthetic speech, there has been a recent increase in activity to develop appropriate tests for the quality of synthetic speech (14). Previously, most tests had focused on measures of intelligibility, but currently available systems easily saturate these tests. Hence, they yield relatively little information concerning the comparative quality of systems. Most studies have found that fricative and nasal sounds are most in need of improvement, but this probably reflects the amount of research devoted to these segments. Preference tests have also been utilized, and direct tests of comprehension have been made. For example, paragraph texts have been prepared, then directly synthesized through the MITalk system, and presented to listeners. Following the presentation of the speech, multiple-choice questions were asked. A control group was visually presented with the text of these paragraphs and asked the same set of questions. After a short learning interval, it was found that the individuals listening to the synthetic speech were able to perform virtually as well on comprehension tests as those who visually read the corresponding text. This is very encouraging for the use of synthetic speech in practical applications and indicates that these techniques can be readily utilized in many situations.

### Implementation

Integrated circuit chips for waveform encoding and decoding have already been discussed as well as the playback of parametrically encoded speech from stored vocabularies in parametric form. Indeed, these implementations are widely used in a variety of contemporary systems. The comprehensive use of full text-to-speech conversion, such as that found in the MITalk system, requires a higher level of computation consisting of two types of processing. A general-purpose microprocessor is used to perform most of the computation, and high-density read-only memories are needed for the morph and letter-to-sound rule lexicons. The generation of the output-speech waveform is currently accomplished by using a digital signal-processing chip. All of this computation can be conveniently performed at comfortable reading-aloud rates between 150 and 200 words per minute. Currently, no commercial system contains all of the refinements present in the MITalk system, but this can be expected as the cost of implementation steadily decreases. In the future, low-cost and compact implementations can be expected that can readily be used as an option in terminals and personal computers. Further extensions will allow such techniques to generate female speech and children's speech as well as speech in several different languages (15). There are no fundamental difficulties to these extensions, but



a great deal of work must be done in order to produce high-quality speech for these sources.

### Conclusion

In this entry the way in which the nature of the application task dictates the kind of speech synthesis to be utilized is shown. In general, the greater the flexibility needed, the more complex the system must be in order to respond to these needs. If the vocabulary is small and the required flexibility is limited, stored speech (or parametrically encoded speech) is a good choice. Increasingly, however, the need for flexibility is paramount, and comprehensive systems for text-to-speech conversion are winning greater acceptance. Even when the vocabulary (at any given time) is not large, it may be changing on a short-term basis. This results in the need for unrestricted text capability. Although there is considerable room for improvement of the speech quality in these systems, they are certainly acceptable for many applications and are becoming increasingly utilized as the cost of implementation decreases.

### BIBLIOGRAPHY

1. J. L. Flanagan, *Speech Analysis, Synthesis, and Perception*, Academic Press, New York, 1972.
2. J. L. Flanagan, and L. R. Rabiner, *Speech Synthesis*, Dowden, Hutchinson, and Ross, Stroudsburg, Pennsylvania, 1973.
3. C. H. Coker, "A Model of Articulatory Dynamics and Control," *Proc. IEEE* **64**(4), 452-459 (1976).
4. N. S. Jayant, and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*, Prentice Hall, Englewood Cliffs, NJ, 1984.
5. J. Makhoul, "Linear Prediction: A Tutorial Review," *Proc. IEEE* **63**(4), 561-580 (1975).
6. J. Holmes, I. Mattingly, and J. Shearme, *Speech Synthesis by Rule, Language and Speech* **7**, 172-143 (1964).
7. S. R. Hertz, Kadin, J., and Karplus, K. J. "The Delta Rule Development System for Speech Synthesis from Text," *Proc. IEEE* **73**(11), 1589-1601 (1985).
8. D. H. Klatt, "The KLATTalk Text-to-Speech Conversion System," *Proc. IEEE Intl. Conference on Acoustics, Speech, and Signal Processing*, 1589-1592 (1982).
9. J. P. Olive, "Speech Synthesis by Rule," in *Speech Communication* **2**, 255-260 (1974).
10. S. J. Young, and F. Fallside, "Speech Synthesis from Concept: A Method for Speech Output from Information Systems," *Journal of the Acoustical Society of America* **66**(3), 685-695 (1979).
11. J. Allen, "Synthesis of Speech from Unrestricted Text," *Proc. IEEE* **64**, 422-433 (1976).
12. J. Allen, M. S. Hunnicutt, and D. H. Klatt, *Proc. IEEE* **73**(11), (1985) (special issue on man-machine communication by speech).
13. J. Allen, ed., *From Text to Speech: The MITalk System*, Cambridge University Press: 1986.
14. D. B. Pisoni, H. C. Nusbaum, and B. G. Greene, "Perception of Synthetic Speech Generated by Rule," *Proc. IEEE* **73**(11), 1665-1676 (1985).
15. R. Carlson, and B. Granstrom, "A Text-to-Speech System Based Entirely on Rules," *Proc. IEEE Intl. Conference on Acoustics, Speech, and Signal Processing*, 686-688 (1976).

### SPEECH UNDERSTANDING

Speech understanding is usually defined as a transduction from an initial acoustic representation of speech to a representation of meaning. For the purposes of practical systems, meaning can be defined operationally as that representation from which actions performed by the system are derived (1). Speech understanding is distinguished from speech recognition (qv), where the goal is to relate an utterance to (a sequence of) unique words in a dictionary. Until the early seventies most research focused on recognition. The five-year ARPA-funded speech project that began at that time made understanding, rather than recognition, the primary research goal. It was felt that a system's ability to respond intelligently to speech was a more meaningful criterion for the evaluation of speech systems. In addition, it was believed that the speech signal was an impoverished source of information, and knowledge of the context of an utterance was essential for its successful recognition and interpretation. Speech-recognition systems based on dynamic programming, pattern-matching (qv) techniques have been developed for utterances that consist solely of isolated words chosen from a small vocabulary, and to a lesser extent, the same techniques have been extended to connected sequences of words (2). However, this approach, which works by finding the best match between variably pronounced words and a vocabulary of stored acoustic templates for words, is less suited to connected speech because the acoustic input in this case cannot be modeled effectively as a simple concatenation of the pronunciations of its constituent lexical items. In connected speech much of the variability that is factored out by pattern matching conveys information useful for both recognizing and interpreting the utterance. Therefore, it is necessary to start with more basic linguistic units than words, such as phonemes (qv) or distinctive features, and to preserve information concerning the timing and duration of the utterance. Once this step is taken, a knowledge-based, rather than pattern-matching, approach to speech processing becomes inevitable because to derive advantage from the recognition of a particular linguistic unit in the signal, it is necessary to know how that unit relates to the rest of the language in question.

Almost by definition, speech-understanding systems (SUSs) operate with connected, phrasal, sentential, or even paragraph-sized chunks of speech because "understanding" isolated words can only mean the essentially trivial process of associating some meaning with each word of the system's vocabulary and accessing this when the word is recognized. Understanding connected speech is a very complex task, and the design of SUSs has been influenced by research in fields as diverse as acoustic signal processing, (neuro)physiology, (psycho)linguistics, and psychology, as well as AI. SUSs can be classified along several dimensions; e.g., number of speakers and dialects accepted or coverage of target language. To date, SUS have been built that understand a handful of speakers of similar dialect, producing a grammatically restricted subset of language with a vocabulary of about a thousand words. Although there are many potential applications for SUSs, their performance and reliability is still too poor for the majority of these to be practical. By contrast, speaker-dependent, isolated word-recognition systems for small vocabularies using whole-word pattern matching have been employed in a variety of applications, such as airline-baggage handling. Nevertheless, it is generally acknowledged that improvements to even this



type of system, such as bigger vocabularies or greater speaker independence, will require a more knowledge-based approach.

### Theoretical Background

The transduction from speech to meaning must be mediated by a variety of components that utilize diverse knowledge sources (KSs) because the speech signal encodes, in a highly compressed and integrated fashion, many different types of information relevant to the recovery of meaning. This knowledge-based approach contrasts with that taken in whole-word template-matching systems; variability in the pronunciation of words in connected speech is no longer seen as a hindrance to pattern matching but rather as an important source of information, e.g., concerning the location of word boundaries (3) or of contextually important (stressed) information in the utterance. Figure 1 illustrates one possible organization for a SUS and the major KSs it requires to function effectively. In this organization SUS information flows upward as each component constructs intermediate representations, encoding (partial) hypotheses about the input, on the basis of the type of knowledge available to it. Acoustic signal processing digitizes the speech at a sampling rate that preserves the acoustic cues relevant to its comprehension. It also transforms the digitized signal in various ways to represent these cues in a form amenable to phonetic decoding (4,5). For example, a spectral analysis will probably be performed and, for each analyzed frame, additional parameters, such as fundamental frequency or spectral center of gravity, computed. The parameterized signal can then be labeled as a discrete sequence of phones by searching for combinations of acoustic features. For example, if the spectral frame contains areas of "fuzziness," i.e., low-amplitude signals spread evenly across the spectrum, the sound is probably part of a fricative such as [f] or [v]. If the frame has a value for fundamental frequency, it must be a voiced fricative such as [v], etc. In addition, each phone is marked with suprasegmental features representing pitch, duration, and amplitude. The acoustic-phonetic transformation is crucial for the effective operation of a SUS but is still one of the least understood aspects of speech processing. It was identified as the chief weakness in the five ARPA-funded SUSs developed in the 1970s (6). Following the acoustic-phonetic transformation, a phonological analysis is performed on the phonetic representation, which identifies the linguistically important distinctions represented in the phonetic representation of the utterance, e.g., levels and locations of stress, intonation contour, syllable structure, and the sequence of phonemes underlying the utterance (7,8). Phonological analysis is essential to lexical access, which is the process of matching the phonetic form of the utterance with the canonical phonemic representations of words in a dictionary to recover the information stored there about their morphological, syntactic, and semantic properties (see Morphology; Parsing; Semantics). It undoes the effects of phonological processes such as assimilation or contraction, which apply in fluent speech; e.g., the words "did" and "you" might be represented in the dictionary as the following sequence of phonemes: /dId/ and /ju/. However, the acoustic-phonetic transformation might recover actual sounds, or phones, such as [dIjə]; to relate this phonetic sequence to the canonical phonemic representations of "did" and "you," it is necessary to recognize that palatalization has occurred at the word boundary, changing [dj] into [j], and that the unstressed vowel of "you" has been reduced to schwa. Similarly, phonological knowledge

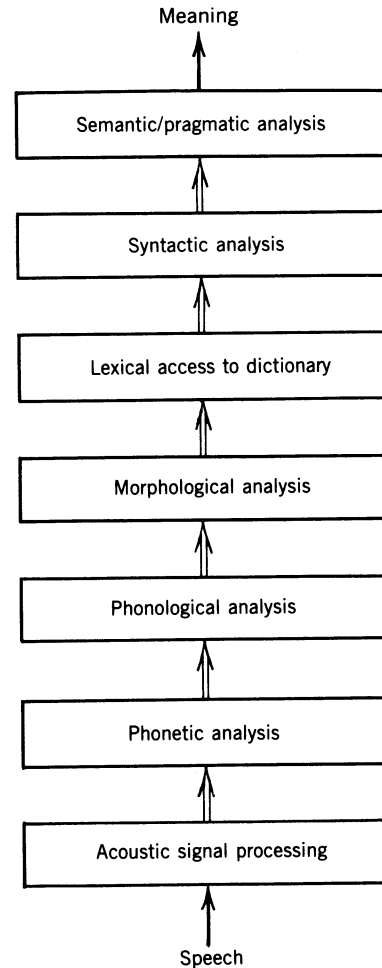


Figure 1. A typical SUS architecture.

concerning allowable sequences of phonemes in syllables, often called phonotactic constraints, can be used to recognize syllable, and hence, word boundaries; e.g., in /hōumhelp/ there must be a boundary between /m/ and the second occurrence of /h/ because no syllable in English can contain /mh/. Comparatively simple information of this type, when combined with lexical knowledge, makes the notoriously difficult task of reliably recognizing word boundaries in the acoustic signal much more tractable.

Once phonological analysis is complete, further processing of the input will be very similar to text understanding (see Natural-language understanding). Processing will be entirely in terms of discrete symbol sets from this point upward through the SUS, and therefore, it is tempting to divide a SUS into a "recognition" phase and an "understanding" phase. However, this view is mistaken because the further morphological, syntactic, semantic, and pragmatic analysis of the utterance contributes to recognition by exploiting more of the redundancy, in the information-theoretic sense, in speech. In some of the ARPA projects, e.g., there was a heavy reliance on syntactic analysis to rule out word hypotheses on the basis of syntactically inadmissible sequences. Before words hypothesized in the speech signal can be matched to lexical entries in the system's dictionary, some morphological analysis will be necessary to relate inflected variants of words to their base forms (9). Apart from regular inflectional morphology, such as plural /s/ or /z/, there are productive derivational morphologi-

cal processes that cannot be dealt with exhaustively by expanding the number of dictionary entries; e.g., there is no principled limit to the number of times "great" can be collocated with "great-grandmother" to produce a new compound noun. After morphological analysis the resulting morphophonemic representation of the speech input can be looked up in the system's dictionary to obtain syntactic and semantic information about the sequence of words hypothesized. Syntactic, semantic, and pragmatic analysis are substantially the same for speech and text understanding. However, there should be interaction between these and lower levels of analysis not only because they will contribute to correct recognition of the utterance but also because aspects of phonological analysis, particularly that relating to stress and intonation, will contribute to its interpretation. Stress, e.g., is relevant to the identification of contextually new information and to finding the correct referents for pronouns. Thus, the degree of integration and interaction between different sources of information in the speech signal prevents any principled distinction between recognition and understanding. Indeed, the separation of the development of the recognition and understanding components of the SUS developed jointly by SRI International and System Development Corporation arguably explains why this major ARPA-funded SUS never worked as an integrated system (10).

This brief description of the contribution of different KSs to speech understanding only covers the major processes; e.g., in a SUS intended to cope with several speakers, variations of voice quality, accent, and dialect must also be dealt with by the acoustic, phonetic, and phonological components. Below, some issues in the design of components for the lower levels of speech analysis are discussed. (See the entries on text understanding for the higher levels.) The KSs deployed in speech understanding are primarily linguistic in nature, and research on them is mainly the concern of linguistic theory. However, the effectiveness of a SUS depends as much on using these KSs efficiently as on developing their content. The major contribution of AI has been to develop techniques for the representation, interpretation, and integration of KSs in a SUS. The task of speech understanding is sufficiently complex to strain the limits of current computing technology. In existing SUSs the generally errorful nature of acoustic-phonetic analysis, and the consequent unreliability of many of the specific hypotheses under consideration by a SUS at any given point, coupled with the frequent genuine ambiguity of speech with respect to any given KS, make issues of system organization and processing strategies crucial for the construction of an effective SUS capable of functioning within practical time and space constraints. Some representative solutions that have been proposed for these problems are discussed in the following sections.

### Acoustic-Phonetic Analysis

Undoubtedly the most crucial area in speech processing in need of more research is acoustic-phonetic analysis. If acoustic-phonetic analysis is errorful, false hypotheses will propagate through a SUS, causing much unnecessary computation and, in the worst case, an incorrect analysis. However, if the initial phonetic representation(s) derived from the acoustic signal could be guaranteed to be unique and correct, the only indeterminacies a SUS would face would be those arising from genuine linguistic ambiguities, most of which are temporary indeterminacies resolvable in terms of further information

available in the speech signal. The segmentation and identification of the acoustic signal into a sequence of linguistic units has proved extremely difficult. First, speech is a code, not a cipher (11); i.e., the acoustic cues associated with segments are not in a one-to-one relationship with those segments; rather these cues are heavily influenced by neighboring segments and so signal the presence of several segments in parallel. For example, the spectral cues to the presence of /d/ in /di/ and /du/ are very different because they are influenced by the following vowel. Moreover, it is not possible to divide the acoustic signal into a /d/ and a following vowel in any motivated manner. These observations prompted the theory that the invariant aspect of these segments is an abstract articulatory target that is not always achieved because of the continuous motion of the vocal tract and led to accounts of speech perception as a process mediated by speech production (11,12). Such analysis-by-synthesis or completely top-down models (see Processing, bottom-up and top-down) would be, however, very computationally expensive since they require that a SUS has the capacity to generate, in principle, all possible utterances and test them against the acoustic input. More recently, it has been argued that acoustic cues to distinctive features (7), as opposed to phonemes or allophones, do contain invariant cues (13), but this claim is controversial (14). Second, acoustic cues are often very minimal in unstressed speech and contexts where there is more redundancy in the speech signal. This often causes many false hypotheses in systems where the acoustic-phonetic component will hypothesize a segment from a fixed inventory, say an allophone or allophones, for every portion of the utterance. An alternative and more attractive approach is not to force an overly specific hypothesis but to iteratively refine the analysis of the acoustic signal from broad to detailed phonetic units as far as the signal allows (15). Thus, false hypotheses will not be propagated through the SUS, although at points the phonetic analysis may lack detail. Third, the acoustic cues to units vary from speaker to speaker because of physiological differences in the vocal tract, differences of characteristic voice quality, etc. (16). Human listeners are able to compensate for these differences rapidly and fluently, but there is still little understanding of how to model this process automatically. Most commercial speech-recognition systems require lengthy training sequences with users repeating each word in the system's vocabulary several times and are therefore very speaker-dependent. In the ARPA projects several of the SUSs developed achieved a degree of speaker independence by attempting to parameterize acoustic-phonetic analysis for a new speaker on the basis of a training sentence the system knew and the user was required to speak. A mapping could then be made between portions of the utterance and a phonetic inventory.

In addition to segmentation into some inventory of units, phonetic analysis must include a representation of the prosodic, suprasegmental aspects of speech, such as stress and intonation. The acoustic cues associated with these phenomena are fundamental frequency, duration, amplitude, and pausing. Reliably measuring fundamental frequency is difficult, as is factoring out the effects of intrinsic fundamental frequency and duration of segments from genuine suprasegmental phenomena in order to recognize stressed syllables, intonational contours, and intonational phrasing. In all of the ARPA project SUSs, suprasegmental acoustic-phonetic analysis was virtually nonexistent and segmental analysis inadequate. The final performance of each system was mainly determined by the effectiveness of higher levels of analysis in

correcting errors at the phonetic level. Thus, constraints on the microworld in which each SUS operated and on the range of constructions accepted were exploited in syntactic and semantic analysis to predict what was being said. More recent systems employ more sophisticated acoustic-phonetic analysis, integrating information from a variety of transformations of the acoustic signal and constructing several types of phonetic representations, but performance is still limited to an average 70% successful recognition of phonemes from utterances produced by a small number of speakers (17).

### Phonological Analysis

Phonology is concerned with the linguistically significant, meaningful patterns of sound in a particular language, including the linguistically significant aspects of suprasegmental, prosodic phenomena. A phonological component is essential for any knowledge-based connected speech-processing system because the system will require knowledge of the phonological processes active in the language, and their domain of application, to recover canonical pronunciations for words that can be matched against a dictionary entry, and to derive further cues to the syntactic and semantic/pragmatic interpretation of the utterance. Phonological components were developed for the ARPA project SUSs and other systems developed during this period (18). However, they were largely restricted to lexical, segmental processes and mostly dealt with phonologically governed variation by generating alternative pronunciations for individual lexical items and storing these in an expanded dictionary. This approach cannot deal adequately with phonological processes that span word boundaries, such as palatalization described above (19). The largest domain of application for a phonological rule is the intonational phrase, which is often coextensive with a full sentence; therefore, phonology cannot be treated in terms of variant pronunciations for lexical items. Because phonological processes are rule-governed and part of the language system, a phonological analysis provides much important information for a SUS; e.g., different types of phonological rule are blocked by different types of linguistic boundaries between segments, so the nonapplication of a phonological rule in an appropriate segmental environment is a clue to the presence of a boundary that blocks its application. As argued above, this is useful to aid segmentation of speech into syllables and words, but it can also provide clues for syntactic analysis; palatalization spans word boundaries but is blocked at the boundaries of major syntactic constituents (20) so its nonoccurrence can be used to resolve an ambiguity concerning the presence of such a boundary at that point in the speech signal. Phonological rules also vary between dialects; therefore, a SUS capable of understanding speakers of different dialects would require knowledge of these differences and an ability to reconfigure itself for their speech. Palatalization, e.g., occurs more frequently and more freely in dialects of American, than British, English.

At the time of the ARPA project, phonological theory was stagnant, and in particular, there was little interest in extending the domain of inquiry beyond segmental processes. However, since the late seventies a number of new approaches to phonology have been developed, such as autosegmental, metrical, and dependency phonology, which take as their central concern suprasegmental phenomena (21). Few of these developments have been incorporated into SUSs, although some have been incorporated into speech-synthesis systems (22,23)

and much of this work is precise and formal enough to be suitable for machine applications. Improvements in the performance of SUSs will certainly require that these developments be incorporated into a much enhanced phonological component that can provide more than variant pronunciations for individual words.

### Knowledge-Source Interpretation

A KS is of no use in a SUS if the knowledge it encodes cannot be represented in a way that allows its interpretation and deployment by machine. The notation employed to represent knowledge in a given field is most naturally determined by the experts in that field of knowledge; e.g., phoneticians typically use the International Phonetic Alphabet for phonetic labeling. However, since choice of representation affects the application of knowledge, the representation systems of KSs in SUSs have often been a compromise between descriptive adequacy and computational efficiency. For instance, in the ARPA project every SUS, with the exception of HWIM (24), employed a syntactic representation thought to be unable to express all of the grammatical possibilities of English. Formal language and automata theory offer efficient algorithms for the application of KSs expressed as sets of rules with the appropriate formal properties (25), and much research on representations for KSs, both in theoretical linguistics and AI, has attempted to develop descriptively adequate notations with these formal properties. For example, minimally augmented context-free notations have been argued to be descriptively adequate for English syntax (26) and phonology (3). Similarly, finite-state transducers have been developed for morphological analysis (27). However, successes of this kind do not lead automatically to computationally tractable KSs since the rule sets required to express knowledge in this form may be extremely large. In addition, it seems unlikely that all KSs employed in a SUS can be expressed within such restricted notations; therefore, more specialized and powerful techniques have also been developed, such as interpreters for production systems (28) (see Rule-based systems) and augmented transition networks (ATNs) (29) (see Grammar, augmented-transition-network). Some expert-system shells (see Expert systems) appear to have promising applications for the acoustic-phonetic transformation because of the more inferential and therefore principled nature of rule application (17) and the ability to factor different aspects of knowledge, which will aid parameterization of the system for different speakers (30). In addition, other AI techniques associated with knowledge representation and text understanding are relevant to interpretation of KSs in a SUS. The better the understanding of a particular domain, the greater the chance of representing that knowledge both adequately and efficiently. Moreover, it is likely that different representation schemes will be most effective for different KSs; therefore, SUS architectures that impose a uniform scheme on all KSs, such as HEARSAY-II (qv) (31) or HARP (qv) (32), are not ideal.

Choice of representation is affected by factors other than the availability of an interpretation technique for a particular scheme; e.g., several SUSs do not attempt to map directly between the acoustic signal and the phonetic alphabet but construct intermediate representations, marking acoustically salient features such as nasality, to aid the process of recognizing individual phones. This reflects the difficulty, discussed above, of relating phones to distinct and invariant sets of acoustic

features and a trend away from pattern matching against a continuous representation of speech toward processing of discrete symbol sets as early as possible in this process. Representations are also affected by the order in which different KSs are brought to bear on the speech signal and the overall architecture of the SUS; recently, it has been proposed that initial phonetic analysis should mark consonants, vowels, and stressed and unstressed syllables and that this simple representation should be used to derive a set of word candidates from a suitably organized dictionary (33). Detailed phonetic analysis would then be applied to the stressed syllable(s) to discriminate between candidates. In a SUS employing this approach, lexical constraints are applied before detailed phonetic analysis; therefore, the role of phonetic analysis is redefined quite radically.

### System Architecture

The bulk of the AI literature on SUSs concerns intercomponent communication during processing. This issue is crucial because ambiguities need to be resolved rapidly to avoid unnecessary computation and because redundancy between KSs can be used to factor out false hypotheses caused by either system errors or genuine ambiguity in the speech signal. For example, the acoustic-phonetic component might hypothesize an aspirated /p/ or /b/ followed by a vowel and /t/, which would result at least in the word candidates "put" and "but." However, it is likely that one of these could be rejected on the basis of syntactic analysis since verbs and conjunctions do not occur in the same syntactic environments. Similarly, there might be a genuine syntactic ambiguity in an utterance, such as "He gave her dog biscuits," in which "her" may be functioning as an adjective or noun. But in this case the ambiguity can be resolved by the different stress and intonation that will accompany the two interpretations. The architectures proposed are basically hierarchical, like that of Figure 1, with a serial flow of information through a chain of component KSs, or heterarchical with no constraint, in principle, on the flow of information between components (34). The advantage of the hierarchical approach is that there appears to be a natural order for the application of KSs to speech input; syntactic analysis can only proceed on the basis of lexical information, etc. Moreover, overall system control is simple. However, there are many occasions when nonserial interactions between the chain of components are useful; e.g., aspects of the prosodic, suprasegmental structure of an utterance will be relevant to its phonological, syntactic, semantic, and pragmatic interpretation. Nonserial interaction can be achieved within the hierarchical model by passing up all of the possible analyses compatible with a given component to the next component, which then selects a subset of analyses, etc. But this only works if the intermediate representations passed up through the SUS are enriched to include all of the information analyzed so far that may be of use to some higher component; thus, the input to the syntactic component, in addition to syntactic information about words, must include all of the available information of potential relevance to syntactic analysis, such as prosodic information, and all information relevant to semantic/pragmatic analysis must be carried through as well. This is likely to strain representation schemes and is computationally expensive because many unnecessary, false hypotheses are computed. These false hypotheses are often avoidable, in principle,

because the disambiguating information is temporally available, encoded in the part of the speech signal already analyzed by lower levels, but in the hierarchical model it is not applied until this input reaches the appropriate component in the serial chain. Heterarchical systems avoid this inefficiency by allowing components to apply in the most efficient order for a given input at the expense of a very complex flow of control within the system and considerable intercomponent communication complexities. Each component must be provided with the means to request and receive information from, or start specific processing in, any other component. This requires specialized communication channels between every component in the system. Developing an adequate control system for such a model may well be impossible because it involves envisaging all possible flows of control at the design stage. Attempts at developing such models have been reduced to human simulation of each component (35). In practice, workable heterarchical models for SUSs have been restricted to uniform representations across KSs and a single global data structure, as in blackboard systems (qv).

Since the ARPA project there has been much interest in SUSs that can be run on parallel machines (36), and the emphasis has been on hierarchical systems that still achieve interaction through selective filtering of hypotheses. Thus, a simple architecture is maintained and more powerful hardware used to cope with the extra computation. One recent major project proposes to use the Chart (37) as a global data structure within a hierarchical architecture, employing as much parallelism as the linearized nature of speech input will allow (30). This approach shares the advantage of a global data structure with the blackboard architecture but does not require a uniform representation scheme across components. None of these proposals represents a theory of nonserial interaction in speech understanding; rather they offer general architectures that attempt to support any interaction that may be required. The designers of the blackboard specifically wanted an architecture capable of supporting arbitrary interactions and thus application to other tasks, such as vision (38). An alternative approach is to specify explicitly the interactions required in a SUS and to develop a specialized architecture capable of supporting them; this approach requires a better understanding of speech than is current but offers the possibility of far more efficient and effective SUS architectures (39).

### Processing Strategies

Various processing strategies have been imposed on different SUS architectures in an attempt to reduce the computation required for successful analysis in the normal case. Both hierarchical and heterarchical systems can operate bottom-up, in an essentially data-driven way, or top-down, using knowledge to produce hypotheses concerning the input (see Processing, bottom-up and top-down). However, most recent SUSs have operated bottom-up because of the rather weak predictability of speech on the basis of the KSs that can currently be deployed effectively in a SUS. Similarly, SUSs can explore the search space in a depth-first or breadth-first manner (see Search; Search, depth-first). Most have operated breadth-first because of the uncertain and errorful nature of many hypotheses but have employed scoring techniques to keep the size of the active search space manageable. One such technique,

shortfall scoring, which involves measuring the summation of individual word candidate scores against a theoretical upper bound for this score and processing the hypothesis with the least difference first, guarantees a SUS will find the best scoring complete hypothesis for the utterance first (40). However, this does not guarantee that the highest ranked hypothesis is correct; the effectiveness of the components that contribute to the generation of word hypotheses is still the crucial factor in the overall performance of the system. The scoring of partial hypotheses throughout a SUS in a more linguistically motivated fashion is extremely difficult. Scores must be carried across components and should reflect the differing contributions of each KS. However, the weight that should be attached to any KS must vary with context; e.g., in the recognition of an unstressed and phonetically reduced preposition, syntactic analysis should be weighted more highly relative to acoustic analysis than in recognition of a stressed syllable. In addition, analyses must be scored through time; an analysis that starts with low scores may end with the highest because redundancy in the information encoded in speech propagates both left and right through the input. Although some scoring schemes that have been used in implemented SUSs do improve performance, this is either for theoretical reasons connected with the scoring technique, as with shortfall scoring, or because they have been developed by trial and error and evaluated solely on the basis of run-time performance, as with the focus-of-attention mechanism in the HEARSAY-II blackboard system. In the former case the technique is useful but limited; in the latter case potentially valuable insights into the task of speech understanding become lost in the scoring technique (41).

Analysis of the speech signal can proceed from left to right through the linearized signal or middle out in both directions from islands of greater acoustic reliability. The island-driven approach has the advantage of taking relatively error-free phonetic data as its starting point at the expense of a more complex control structure and system organisation, as in HWIM (40). Human listeners appear to pay greater attention to stressed syllables (42), which are generally more clearly enunciated and therefore more easily analyzed phonetically. In addition, the phonological structure of English vocabulary is constrained in such a way that a unique word can usually be derived from a crude phonetic analysis of syllable structure coupled with detailed analysis of its stressed syllable (33). Therefore, the island-driven approach is essentially correct, although it would be more effective if processing began at stressed syllables by explicitly searching for them rather than at arbitrary high-scoring portions of the acoustic signal. A related approach that avoids the extra overheads in HWIM caused by middle-out analysis is to use an appropriate scoring strategy in a left-to-right system, which is able to take account of forthcoming speech events in the right context of the existing partial hypothesis (43).

### Current Trends

Since the ARPA project in the seventies there has been a period of problem-oriented, rather than system-building, research in speech understanding. Much of this research has focused on the acoustic-phonetic transformation as a result of new evidence demonstrating the informational richness of the acoustic signal (44). Now there is renewed interest in building complete systems (30) incorporating this research and re-

newed concern with issues such as system architecture. However, the majority of the knowledge-based systems that are being developed are restricted to continuous speech recognition rather than understanding. Improvements in acoustic-phonetic analysis suggest that higher levels of analysis are not crucial for recognition of continuous speech, contrary to prevailing opinion at the time of the ARPA project. In addition, the problems of understanding, such as knowledge-representation (qv) issues, the restriction to a microworld, etc., remain unsolved.

### Systems

The main SUSs developed in the ARPA project were HARPY, HWIM, HEARSAY-II, and SRI/SDC. HARPY came closest to the performance criteria specified for the project (45). However, HARPY's architecture required precompilation of all KSs into a single finite-state network so the language accepted by the system was more restricted than that for the other systems (41). Each of these systems is briefly described by its chief designer in Ref. 46 and they are evaluated in Refs. 6 and 10. A system of the same period developed at IBM but with improved performance over HARPY on the same subset of English is described in Ref. 47. The HEARSAY-II system is reimplemented as a production system in Ref. 28, and extensions and improvements to the original blackboard architecture are described in Ref. 41. Several SUSs have been developed for European languages, such as KEAL (48) and MYRTILLE-II (49) for French and EVAR (50) for German. However, these systems have not surpassed the ARPA project systems in performance or design. An automated airline reservation system that incorporates continuous speech understanding is described in Ref. 51. This system, developed at Bell Laboratories, conducts a dialogue over a telephone to establish the appropriate reservation. It employs whole-word template-matching techniques to recognize words from a 127-word vocabulary but relies on semantic constraints deriving from this very restricted task domain and syntactic constraints imposed by a restrictive finite-state grammar to achieve robust performance.

### Further Reading

The best introductions to speech understanding are Refs. 46 and 52. Two further more recent collections are Refs. 53 and 54, and Ref. 55 provides an up-to-date, advanced course. Articles on the higher levels of speech understanding can be found in the journals *Artificial Intelligence* and *Computational Linguistics*. Issues relevant to acoustic-phonetic analysis are dealt with in *Journal of the Acoustical Society of America*, *Journal of Phonetics* and *Language and Speech*. Many articles on whole-word template-matching techniques for isolated and connected speech and on speech signal processing can be found in the journal of the Institute of Electrical and Electronic Engineering, Acoustics, Speech and Signal Processing. Articles on the application of speech-processing systems can be found in *Speech Technology*, and *Human Factors* sometimes contains performance evaluations of systems. The *International Journal of Man-Machine Studies* also publishes articles on speech processing. Many conference proceedings also contain relevant articles, such as the *International Joint Conference on Artificial Intelligence*, *Coling*, *Association of Computational Lin-*

guistics, *Acoustic Society of America, International Conference on Acoustics, Speech and Signal Processing*, and others.

## BIBLIOGRAPHY

1. A. Newell, A Tutorial on Speech Understanding Systems, in Ref. 52 pp. 3–54.
2. L. R. Rabiner and S. E. Levinson, "Isolated and connected word recognition—theory and selected applications," *IEEE Trans. Commun. Com-29*(5), 621–659 (1981).
3. K. W. Church, *Phrase Structure Parsing: A Method for Taking Advantage of Allophonic Constraints*, Indiana University Linguistics Club, Bloomington, IN, 1983.
4. J. L. Flanagan, *Speech Analysis, Synthesis and Perception*, 2nd ed., Springer-Verlag, New York, 1972.
5. L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, NJ, 1978.
6. D. H. Klatt, "Review of the ARPA speech understanding project," *J. Acoust. Soc. Am.* **62**, 1345–1366 (1977).
7. R. Lass, *Phonology*, Cambridge University Press, Cambridge, U.K., 1984.
8. D. R. Ladd, *The Structure of Intonational Meaning*, Indiana University Press, Bloomington, IN, 1980.
9. P. H. Matthews, *Morphology*, Cambridge University Press, Cambridge, U.K., 1974.
10. A. Barr and E. A. Feigenbaum, *The Handbook of Artificial Intelligence*, Vol. I, Kaufmann, Los Altos, CA 1981.
11. A. M. Liberman, F. S. Cooper, D. P. Shankweiler, and M. Studert-Kennedy, "Perception of the speech code," *Psychol. Rev.* **74**, 431–461 (1967).
12. M. Halle and K. N. Stevens, Speech Recognition: A Model and a Program for Research, In J. A. Fodor and J. J. Katz (eds.), *The Structure of Language*, Prentice-Hall, Englewood Cliffs, NJ, pp. 604–612, 1964.
13. K. N. Stevens and S. E. Blumstein, "Invariant cues for place of articulation in stop consonants," *J. Acoust. Soc. Am.* **64**, 1358–1368 (1978).
14. L. Lisker, "The pursuit of invariance in speech signals," *J. Acoust. Soc. Am.* **77**, 1199–1202 (1985).
15. S. R. Johnson, J. H. Connolly, and E. A. Edmonds, Spectrogram Analysis: A Knowledge-based Approach to Automatic Speech Recognition, In M. A. Bramer (ed.), *Research and Development in Expert Systems*, Cambridge University Press, Cambridge, U.K., 1985.
16. J. Laver, *The Phonetic Description of Voice Quality*, Cambridge University Press, Cambridge, U.K., 1980.
17. R. De Mori, P. Laface, G. Petrone, and M. Segnan, Access to a Large Lexicon Using Phonetic Features, *Proceedings of EUSIPCO, Erlangen*, 1983.
18. P. S. Cohen and R. L. Mercer, The Phonological Component of an Automatic Speech Recognition System, in Ref. 52 pp. 275–320.
19. D. H. Klatt, Speech Perception: A Model of Acoustic-Phonetic Analysis and Lexical Access, in Ref. 53 pp. 243–288.
20. W. E. Cooper and J. Paccia-Cooper, *Syntax and Speech*, Harvard University Press, Cambridge, MA, 1980.
21. N. Smith and H. van der Hulst (eds.), *The Structure of Phonological Representations*, Foris, Dordrecht, Holland, 1982.
22. J. B. Pierrehumbert, "Synthesising intonation," *J. Acoust. Soc. Am.* **70**, 985–995 (1981).
23. B. J. Williams, A Metrical Algorithm for Lexical Stress Assignment in English, *Proceedings of 109th Meeting of Acoustic Society of America*, Austin, TX, p. S39, 1985.
24. W. A. Woods, Control of Syntax and Semantics in Continuous Speech Understanding, in Ref. 54 pp. 337–364.
25. A. V. Aho and J. Ullman, *The Theory of Parsing, Translating and Compiling*, Vol. I, Prentice-Hall, Englewood Cliffs, NJ, 1972.
26. G. J. M. Gazdar, G. K. Pullum, I. A. Sag, and E. Klein, *Generalized Phrase Structure Grammar*, Blackwell, Oxford, 1985.
27. K. Koskenniemi, A General Computational Model for Word-Form Recognition and Production, *Proceedings of Coling84*, Stanford, CA, July 1984, pp. 178–181.
28. D. L. McCracken, *A Production System Version of the Hearsay-II Speech Understanding System*, UMI Research, Ann Arbor, MI, 1981.
29. W. A. Woods, Syntax, Semantics and Speech, in Ref. 52, pp. 345–400.
30. H. Thompson, Speech Transcription: An Incremental Interactive Approach, *Proceedings of the European Conference on AI*, Pisa, Italy, September 1984, Elsevier Science, New York, pp. 697–704, 1984.
31. L. D. Erman and V. R. Lesser, The Hearsay-II Speech Understanding System: A Tutorial, in Ref. 46, pp. 361–381.
32. B. T. Lowerre and D. R. Reddy, The Harpy Speech Understanding System, in Ref. 46, pp. 340–360.
33. D. P. Huttenlocher and V. W. Zue, A Model of Lexical Access from Partial Phonetic Information, *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, San Diego, CA, March 1984, pp. 2641–2644.
34. D. R. Reddy and L. D. Erman, Tutorial on System Organisation for Speech Understanding, in Ref. 52, pp. 457–480.
35. W. A. Woods and J. Makhoul, Mechanical Inference Problems in Continuous Speech Understanding, *Proceedings of the Third International Joint Conference Artificial Intelligence*, Stanford, CA, pp. 200–207, 1973.
36. S. E. Fahlmann, G. E. Hinton, and T. J. Sejnowski, Massively Parallel Architectures for AI: NETL, Thistle and Boltzmann Machines, *Proceedings of the Third National Conference for Artificial Intelligence*, Washington, DC, pp. 109–113, 1983.
37. M. Kay, The MIND System, in R. Rustin (ed.), *Natural Language Processing*, Algorithmics, New York, pp. 155–188, 1973.
38. L. D. Erman and V. R. Lesser, A Multi-Level Organisation for Problem Solving Using Many, Diverse, Cooperating Sources of Knowledge, *Proceedings of the Fourth International Joint Conference of AI*, Tbilisi, Georgia, pp. 483–490, 1975.
39. E. J. Briscoe and B. K. Boguraev, Control Structures and Theories of Interaction in Speech Understanding Systems, *Proceedings of Coling84*, Stanford, CA, pp. 259–266, 1984.
40. W. A. Woods, "Optimal search strategies for speech understanding control," *Artif. Intell.* **18**, 295–326 (1982).
41. B. Hayes-Roth, A Blackboard Model of Control, Report No. HPP-83-38, Department of Computer Science, Stanford University, 1983.
42. A. Cutler and D. Norris, Syllable Boundaries and Stress in Speech Segmentation, *Proceedings of the 109th Meeting of Acoustic Society of America*, Austin, TX, pp. S39, 1985.
43. A. M. Johnstone and G. Altmann, Automated Speech Recognition: A Framework for Research, Research Report No. 233, Edinburgh University, Department of AI, 1984.
44. R. A. Cole, A. I. Rudnicky, V. W. Zue, and D. R. Reddy, Speech as Patterns on Paper, in Ref. 53, pp. 3–50.
45. A. J. Newell, J. Barnett, J. W. Forgie, C. Green, D. H. Klatt, J. C. R. Licklider, J. Munson, D. R. Reddy, and W. A. Woods, *Speech Understanding Systems: Final Report of a Study Group*, North-Holland, Amsterdam, 1973.
46. W. A. Lea (ed.), *Trends in Speech Recognition*, Prentice-Hall, Englewood Cliffs, NJ, 1980.



47. L. R. Bahl, J. K. Baker, P. S. Cohen, N. R. Dixon, F. Jelinek, R. L. Mercer, and H. F. Silverman, Preliminary Results on the Performance of System for the Automatic Recognition of Continuous Speech, *International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, PA, pp. 512–514, 1976.
48. G. Mercier, A. Nouhen, P. QUINTON, and J. Siroux, The KEAL Speech Understanding System, in Ref. 54, pp. 525–545.
49. J. M. Pierrel and J. P. Haton, The MYRTILLE-II Speech Understanding System, in Ref. 54, pp. 553–570.
50. H. Niemann, The Erlangen System for Recognition and Understanding of Continuous Speech, in J. Nehmer (ed.), Springer Verlag, Berlin, pp. 330–348, 1982.
51. S. E. Levinson and K. L. Shipley, "A conversational-mode airline information and reservation system using speech input and output," *Bell Sys. Tech. J.* **59**, 119–137 (1980).
52. D. R. Reddy (ed.), *Speech Recognition*, Academic, New York, 1975.
53. R. A. Cole (ed.), *Perception and Production of Fluent Speech*, Erlbaum, Hillsdale, NJ, 1980.
54. J. C. Simon (ed.), *Spoken Language Generation and Understanding*, Reidel, Dordrecht, Holland, 1980.
55. F. Fallside and W. A. Woods (eds.), *Computer Speech Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

E. J. BRISCOE  
University of Cambridge

## STEREO VISION

Intelligence can be defined as the ability of an entity to structure its interactions with its environment so as to achieve its goals. The ability to acquire and maintain an environmental model is thus an essential characteristic of an intelligent system. In this entry, methods are described for recovering geometric descriptions of visible surfaces and objects within a particular paradigm called computational stereo. The computational stereo paradigm involves a sequence of explicit algorithmic steps that are "mechanical" in nature and that do not require logical inference or reference to prior knowledge. This approach leads to adequate performance in many cases, but in the more complex situations mechanical approaches to stereo reconstruction will fail unless augmented by reasoned analysis and stored knowledge.

The human visual ability to perceive depth is both commonplace and puzzling. One perceives three-dimensional spatial relationships effortlessly, but the means by which one does so are largely hidden from introspection. Binocular stereopsis, however, is one method for depth perception that is relatively well-understood. It allows one to recover information about the three-dimensional shapes and locations of objects by comparing two images of a scene from different perspectives. Stereo is an attractive source of information for machine perception as well because it leads to direct range measurements and, unlike monocular approaches, does not merely infer depth or orientation through the use of photometric and statistical assumptions. Once two stereo images are brought into point-to-point correspondence, recovering range values by trigonometric means is relatively straightforward. Another advantage is that stereo is a passive method. Although ranging methods that use structured light, laser rangefinders, or other active sensing techniques are useful in tightly controlled domains, such as industrial automation applications, they are unsuitable for more general environments.

Research concerned with the three-dimensional geometric reconstruction of scenes from images can be divided into three approaches: those methods that use range information directly provided by an active sensor; those methods that use only monocular information available in a single image (or perhaps several images, under different lighting, but from a single viewpoint); and those methods that use two or more images taken from different viewpoints, perhaps at different times. In this entry the third approach is referred to as "generalized stereo." The generalized-stereo paradigm includes conventional stereo as well as what is often called optical flow. The conventional-stereo technique is to record two images, simultaneously or sequentially, by using laterally displaced cameras (Fig. 1). The optical-flow technique is to record two or more images sequentially, usually with a single camera moving along an arbitrary path. Conventional stereo and optical flow have similar geometric foundations but use considerably different mathematical formulations. Optical flow is not discussed in this entry (see Optical flow).

The most common use of computational stereo is in the interpretation of aerial images. Survey papers by Konecny and Pape (1) and Case (2) describe the state of the art of automated stereo in this field. Fritz (3) reports on a panel discussion on automated correlation of stereo data that covered the important problems and current research directions in this area. Other applications include passive visual navigation for autonomous vehicle guidance, industrial automation, and the interpretation of microstereophotographs for biomedical applications. Each domain has different requirements that affect the design of a complete stereo system.

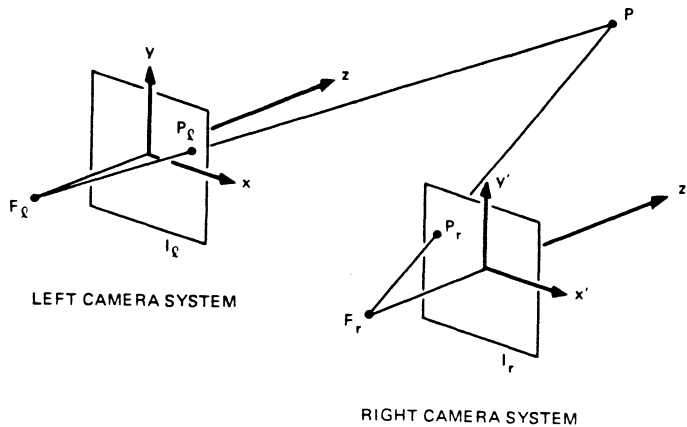
### Computational Stereo Paradigm

Research on computational solutions for the generalized-stereo problem has followed a single paradigm, both in method and intent. The paradigm involves the following steps:

- image acquisition,
- camera modeling,
- feature acquisition,
- image matching,
- distance (depth) determination, and
- interpolation.

**Image Acquisition.** Stereoscopic images can be acquired in a large variety of ways. For example, they may be recorded simultaneously or at time intervals of any duration. They may be recorded from viewing locations and directions that are only slightly different or that are radically different. The most important factor affecting image acquisition is the specific application for which the stereo computation is intended. Three applications have received the most attention: the interpretation of aerial photographs for automated cartography, guidance and obstacle avoidance for autonomous vehicle control, and the modeling of human stereo vision.

Aerial photo interpretation usually involves low-resolution images of a variety of terrain types (4). Aerial stereo images may be either vertical, in which the cameras point directly downward, or oblique, in which the cameras are intentionally directed between the horizontal and vertical directions. Vertical stereo images are easier to compile into precise carto-



**Figure 1.** Stereo. Two camera systems are shown. The focal points are at  $F_L$  and  $F_R$ , the image planes are  $I_L$  and  $I_R$ , and the principal axes are  $Z$  and  $Z'$ . A point  $P$  in the three-dimensional scene is projected onto  $P_L$  in the left image and onto  $P_R$  in the right image. The disparity of  $P$  is the difference in the positions of its projections onto the two stereo image planes. The disparity of  $P$  depends on its location in the scene and on the geometric relation between the camera systems. In most cases the location of  $P$  can be determined from its stereo projections.

graphic measurements, but oblique stereo images cover more terrain and require less stringent control of the aircraft carrying the camera.

Stereo for autonomous vehicle control has been studied in two contexts: as a passive navigation aid for robot aircraft (5) and as part of a control system for surface vehicles (6–8). The images used for aircraft navigation are similar to the aerial photographs used for cartography, except that multispectral sensors are often employed. The images used for surface vehicle control, though, are quite different: they are horizontal, comparatively high-resolution images.

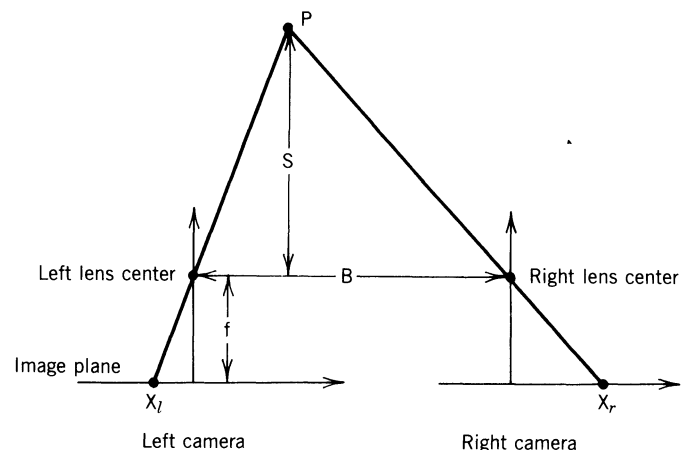
Research on computational models of human stereo vision (qv) has largely employed synthetic random-dot stereograms for experimental investigation (9–11). A random-dot stereogram consists of two synthetic images of uncorrelated dots, which are constructed to depict perspective views of the same virtual surface (12). Each image by itself contains no depth information because it consists only of random dots. When the pair is viewed stereoscopically, however, the three-dimensional virtual surface is readily perceived. Random-dot stereograms are historically important because they demonstrated conclusively for the first time that human stereo vision does not depend on monocular shape perception. Because the parameters of random-dot stereograms, such as noise and dot density, can be controlled, they allow systematic comparison of human and machine performance. Experiments on human stereo vision using natural imagery instead of random-dot stereograms have also been performed (e.g., see Ref. 11).

Different stereo applications often involve different kinds of scenes. Perhaps the most significant and widely recognized difference in scene domains is between scenes containing cultural features, such as buildings and roads, and those containing only natural objects and surfaces, such as mountains, flat or "rolling" terrain, foliage, and water. Important stereo applications range over both domains. Low-resolution aerial imagery, e.g., usually contains mostly natural features, although cultural features are sometimes found. Industrial applications, on the other hand, tend to involve man-made, cultural objects exclusively. Cultural features present special prob-

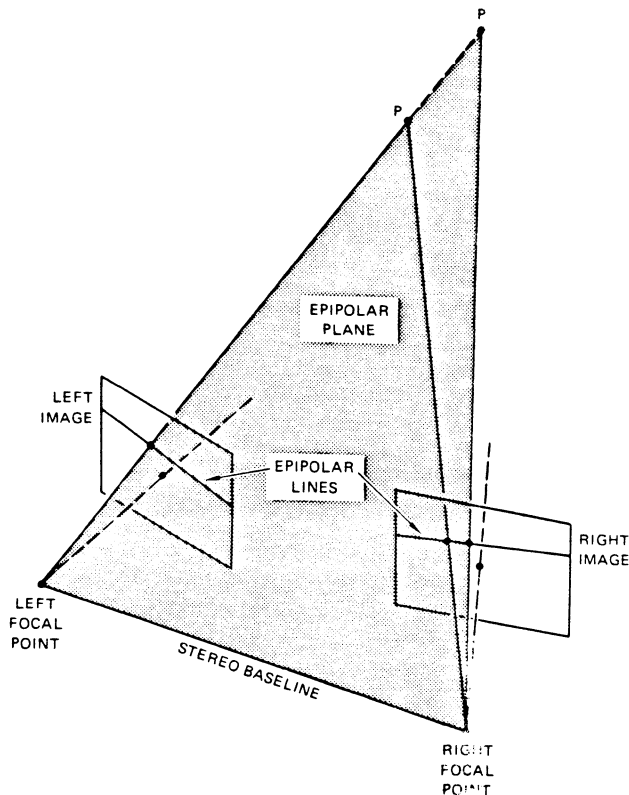
lems. For example, periodic structures such as the windows of buildings and road grids can confuse a stereo system. The relative abundance of occlusions in a city scene can also cause problems because large portions of the images may be unmatched. Cultural objects often have large surfaces with nearly uniform albedo that are difficult to match because of the lack of detail. Stereo systems that have been described in the literature are often intended for specific scene domains, and there has seldom been any attempt to validate the methods in other domains.

**Camera Modeling.** The key problem in stereo depth modeling is to find corresponding points in the two stereo images. Corresponding points are the projections of a single point in the three-dimensional scene. The difference in the positions of two corresponding points in their respective images is called "disparity." Disparity is a function of both the position of the point in the scene and of the position, orientation, and physical characteristics of the stereo cameras. When these camera attributes are known, corresponding image points can be mapped into three-dimensional scene locations (see Fig. 2). A camera model is a representation of the important geometric and physical attributes of the stereo cameras. It has a relative component, which relates the coordinate system of one camera to the other, and it may also have an absolute component, which relates one of the camera coordinate systems to the fixed coordinate system of the scene.

In addition to providing the function that maps pairs of corresponding image points onto scene points, a camera model can be used to constrain the search for corresponding image points to one dimension (Fig. 3). Any point in three-dimensional world space, together with the centers of projection of two camera systems, defines a plane (called an "epipolar" plane). The intersection of an epipolar plane with an image



**Figure 2.** Depth from stereo. This diagram shows how the position of a point can be determined by two projections in a simple case. Left and right cameras with focal distance  $f$  are separated by stereo baseline  $B$  and form coplanar images. One can easily derive the coordinates of a point  $P$  in space from similar triangles. Let  $\Delta = x_r - x_l$  be the disparity of  $P$ . Then  $P = (-x_r B / \Delta, f B / \Delta + f)$  in the coordinate system of the left camera. The distance of  $P$  from the left camera center,  $S = f B / \Delta$ , is proportional to the focal distance and the length of the baseline and is inversely proportional to the disparity. Longer baselines and focal distances lead to more accurate depth measurements for a given image resolution, since  $d\Delta/dS = -fB/S^2$ ; i.e., increasing  $f$  or  $B$  causes disparity to change more rapidly with depth.



**Figure 3.** The epipolar constraint. Left and right camera systems are shown. The line connecting the focal points of the camera systems is called the stereo baseline. Any plane containing the stereo baseline is called an epipolar plane. Suppose a point  $P$  in the scene is projected onto the left image. Then the line connecting  $P$  and the left focal point, together with the stereo baseline, determines a unique epipolar plane. The projection of  $P$  in the right image must therefore lie along the line that is the intersection of this epipolar plane with the right image plane. (The intersection of an epipolar plane with an image plane is called an epipolar line.) If the geometric relationship between the two camera systems is known, we need only search for a match along the epipolar line in the right image.

plane is called an epipolar line. Every point on a given epipolar line in one image must correspond to a point on the corresponding epipolar line in the other image. The search for a match of a point in the first image may therefore be limited to a one-dimensional neighborhood in the second image plane, as opposed to a two-dimensional neighborhood, with an enormous reduction in computational complexity.

When the stereo cameras are oriented such that there is only a (known) horizontal displacement between them, disparity can only occur in the horizontal direction and the stereo images are said to be "in correspondence." When a stereo pair is in correspondence, the epipolar lines are coincident with the horizontal scan lines of the digitized pictures—enabling matching to be accomplished in a relatively simple and efficient manner. Stereo systems that have been primarily concerned with modeling human visual ability have employed this constraint (10,13). In practical applications, however, the stereo pair rarely is in correspondence. In aerial stereo photogrammetry (the process of making measurements from aerial stereo images), e.g., the camera may typically be tilted as much as two to three degrees from vertical (4). With any tilt, points on a scan line in one image will not fall on a single scan line in the second image of the stereo pair, and thus, the com-

putational cost to employ the epipolar constraint will be significantly increased. It is possible, however, to reproject the stereo images onto a common plane parallel to the stereo baseline such that they are in correspondence.

The difference in position and orientation of two stereo cameras is called the relative camera model. Relative camera models are required for depth determination and for exploiting the epipolar constraint. In most cases considerable a priori knowledge of the relative camera model is available, but it is often less accurate than desired. Gennery (14) has developed a method for finding the relative camera model (in terms of differences in azimuth, elevation, pan, tilt, roll, and focal length) from a few well-distributed matches. The method is a least-squares minimization of the errors of the distances of points in image 2 from their predicted locations, as determined by their positions in image 1 through the use of the assumed relative camera model. The nonlinear optimization problem is solved by iterating on a linearization of the problem. In practice, the method has been found to require a good starting point to ensure convergence (5). The minimum number of corresponding points that is necessary to derive a unique relative camera model is 5, but the solution involves five simultaneous third-order equations (15). If as many as eight corresponding points are available, the relative camera model can be obtained directly as the solution of a system of linear equations (16–18). Ganapathy (19) and Strat (20) have described methods for extracting the camera parameters from the transform matrix.

Fischler and Bolles (21) have provided a number of results with respect to the minimum number of points needed to obtain a solution to the camera-modeling problem, given a single image and a set of correspondences between points in the image and the points' spatial (geographic) locations; they also provide a technique for finding the complete camera model even when the given correspondences contain a large percentage of errors. Although this work was directed at the problem of establishing a mapping between an image and an existing database of geographic locations, it is possible to apply the result to the stereo problem. In fact, tying the stereo pair to an existing database offers the possibility of employing scene-dependent constraints beyond those available from the imaging geometry. They also presented a technique for fitting a model to experimental data, called RANSAC, or Random-Sample Consensus. This method is radically different from conventional methods, such as least-squares minimization, which begin with large amounts of data and then attempt to eliminate invalid points. RANSAC uses a small, randomly chosen set of points and then enlarges this set with consistent data when possible. This strategy avoids the problem, common to least-squares and similar methods, of a few gross errors, or even a single one, leading to very bad solutions. In practice, RANSAC can be used as a method for selecting and verifying a set of points that can be confidently fit to a model with a conventional method (such as least-squares minimization).

Camera modeling can be extended to include distortions introduced by the image formation process. Significant image distortion will degrade the accuracy of depth measurements unless corrected. Two kinds of image distortion are commonly found—radial and tangential. Radial distortion causes image points to be displaced radially from the optical axis (the axis through the centers of curvature of the lens surface), whereas tangential distortion is caused by imperfect centering of lens elements, resulting in image displacements perpendicular to the radial lines. Moravec described a method to correct for

distortion using a square pattern of dots (6). His method finds fourth-degree polynomials that transform the measured positions of the dots and their neighborhoods to their nominal positions.

There is an important caution to the discussion of camera modeling. Techniques that are analytically adequate may be numerically unstable, and may fail to provide useful results in real problem domains.

**Feature Acquisition.** Matching is required in both the camera-modeling and the depth-determination steps of the stereo process; however, featureless areas of nearly homogeneous brightness cannot be directly matched. Accordingly, most work in computational stereo has included some form of selective feature detection, which is closely coupled with the matching strategy used.

Approaches that apply area matching often use an "interest operator" in an attempt to locate places in one image that can be successfully matched to corresponding points in the second image of a stereo pair. One type of interest operator selects areas that have high image-intensity variance. These areas will not make good features, however, if their variance is due only to brightness differences in the direction perpendicular to the epipolar line. These areas can be culled by demanding that the two-dimensional autocorrelation function have a distinct peak (22). A widely used interest operator is the Moravec operator (6), which works as follows: for each point in one image the variance between adjacent pixels in four directions over a square ( $3 \times 3$ -pixel) neighborhood centered on the point is computed; the minimum variance is selected as the point's interest measure; and points that have a locally maximum interest measure are chosen. Intuitively, each chosen point must have relatively high variance in several directions and must be more "interesting" than its immediate neighbors.

Feature detection is also required by those approaches that directly match features in the stereo images (as opposed to those that simply use the features to choose areas for correlation matching). The features may vary in size, direction, and dimensionality. Precisely delineated or bounded features are good candidates for matching when the camera model is unknown and the matches are not constrained to epipolar lines because, unlike unbounded linear features, such features have points that can be unambiguously located in the image. Linear features, on the other hand, must be oriented across the epipolar lines if they are to be matched accurately. Another advantage of pointlike features is that they can be matched without concern for perspective distortion. In area-correlation approaches matches of pointlike features are often used to obtain the camera model prior to more extensive matching employed for depth determination. The autocorrelation of local intensity values around a point can be used to establish initial confidences of pointlike feature matches in a way similar to the scores obtained in area cross-correlation (23).

If the camera model is known or derived in a preliminary step, edge elements can be used as primitive matching features (24,25,10). Many distinct models have been proposed as the basis for edge-detecting algorithms. Zero crossings in the Laplacian of a Gaussian-smoothed image [the  $\nabla^2 G$  operator (13)] are often used for this purpose. A difference-of-Gaussians filter is commonly used as a close approximation to the  $\nabla^2 G$  operator.

Mayhew and Frisby (26) argue that zero crossings alone are inadequate as primitive features for matching because they do

not allow the computation of disparity that depends on luminance variations between the zero crossings. They present examples that demonstrate this and propose the use of peaks and troughs as additional primitive features. A major justification for the use of zero crossings is Logan's theorem (27), which establishes that the convolution of an image with a one-octave bandpass filter (such as a Laplacian of a Gaussian) can be recovered from its zero crossings alone. Logan's theorem applies only to one-dimensional signals, however, and its validity for two-dimensional signals is an open question. However, even if in principle the peaks and troughs could be recovered from a reconstructed intensity image, this would be wasteful when the intensity image is already available.

Typically, an edge-detecting algorithm (see Edge detection) produces both a magnitude (that is related to the contrast across the edge) and a direction. In the case of "strong" (i.e., high-magnitude) edges, most of the algorithms yield similar results for operators with comparable mask sizes. Often the same underlying model appears in different implementations. For example, zero crossings in the second derivative are equivalent to local maxima in the first derivative, and most of the conventional edge-detection methods search for approximations to maxima of the first derivative of image intensity. More important are the issues governing the conditions under which "weak" edges are reliable features for matching. Extent, direction, and magnitude of edges have been used as features in making match decisions, but their relative merits have not been established.

For the most part the features that have been used for determining stereo correspondence have been "low level" in the sense that they depend only on local intensity patterns. There are, however, some exceptions. Ganapathy, e.g., described a system for matching vertices in blocks-world stereo scenes across very large viewing angles (28). More recently, Herman and Kanade (29) have developed stereo techniques that match comparatively high-level features such as line junctions (e.g., L, arrow, and fork junctions), explicitly accounting for the way junctions change appearance between views. High level features are more difficult to find, but their use may be necessary for complex cultural scenes.

**Image Matching.** Image matching is a core area in scene analysis and is not covered in full detail here. Instead, this section focuses on those portions of the image-matching problem that are directly relevant to stereo modeling. Features that distinguish stereo-image matching from image matching in general are the following.

The important differences in the stereo images are due to the different viewpoints and not, e.g., to changes in the scene. Therefore, a match is sought between two images, as opposed to a match between an image and an abstract model.

Most of the significant changes will occur in the appearance of nearby objects and in occlusions. Additional changes in both geometry and photometry can be introduced in the film-development and scanning steps but can usually be avoided by careful processing.

Stereo modeling generally requires that, ultimately, dense grids of points be matched.

Ideally, one would like to find the correspondences (i.e., the matched locations) of every individual pixel in both images of

a stereo pair. However, it is obvious that the information content in the intensity value of a single pixel is too low for unambiguous matching. In practice, therefore, coherent collections of pixels are matched. These collections are determined and matched in two distinct ways.

**Area Matching.** Continuous areas of image intensity are the basic units that are matched. This approach usually involves some form of cross-correlation to establish correspondences.

**Feature Matching.** "Semantic features" (with known physical properties and/or spatial geometry) or "intensity anomaly features" (isolated anomalous intensity patterns not necessarily having any physical significance) are the basic units that are matched (see the discussion in the preceding section on feature acquisition.) Semantic features of the generic type include occlusion edges, vertices of linear structures, and prominent surface markings; domain-specific semantic features might include such features as the corner or peak of a building or a road-surface marking; and intensity anomaly features include zero crossings and image patches found by the Moravec interest operator (see Feature extraction).

**Correlation Matching.** Panton (30) describes a cartographic system that uses correlation for stereo matching. Points in a regularly spaced grid in the left image are matched to grid points in the right image. Matching is accomplished by searching along the corresponding epipolar line in the right image for a maximum score for a correlation patch, which is warped to account for predicted terrain relief (estimated from previous matches). Subpixel matches are obtained by fitting a quadratic to the correlation coefficients and picking the interpolated maximum. "Tuning parameters" may be dynamically altered to adapt the system to sensor and terrain variations. Tuning parameters include grid sizes; patch size and shape; number of correlation sites along the search segment; and reliability thresholds for the correlation coefficient, standard deviation, prediction function range, etc. Their intent is to choose the smallest feasible patch subject to the need to compensate for noise and lack of intensity variation in the image. A problem with correlation matching is that the patch size must be large enough to include enough intensity variation for matching but small enough to avoid the effects of projective distortion. [Recent work by Barnard (31) shows that individual pixels can be matched by using a smoothness constraint in the context of a global optimization approach called simulated annealing.] The reliability of matching is continuously monitored to signal when parameters become inappropriate or when the photometry prevents valid matching.

Hannah (32) describes a complete stereo system that implements all the basic steps in the computational stereo paradigm. The system has a modular design that is intended to test alternative methods for implementing the various steps. It uses a hierarchy of stereo images, with each image reduced by a factor of 2 with respect to its predecessor. An interest operator is used to locate a well-scattered collection of points in one high-resolution image, which are then matched (using correlation) through the other image hierarchy. The plausibility of these matches is checked by again performing the search, but in reverse (i.e., beginning with the other high-resolution image). These matched points, called "anchor points," are used to derive a relative camera model (assuming this model has not

already been provided by more accurate calibration procedures), thereby establishing the epipolar constraint. Dense matching is performed using both the epipolar constraint and the anchor points to estimate disparities. A final step interpolates over "holes" in the dense disparity map and, if absolute camera parameters are provided, produces a terrain map.

**Feature Matching.** One of the more severe problems for correlation matching is that large image areas of homogeneous or only slightly varying luminance cannot be matched. Feature-matching techniques take the approach of ignoring these areas and concentrating on matching discrete image structures that represent localized features of the scene. Features that can be precisely located are typically matched in a preliminary stage to establish a relative camera model (e.g., see Ref. 24). A precise camera model to establish the epipolar constraint is necessary before linear features can be matched [but see Hildreth (33) for an example of a way to match contours in the absence of a camera model].

The influential Marr-Poggio stereo model provides a good example of linear feature matching (13,10,11); i.e., the zero crossings obtained by convolving the image with difference-of-Gaussians bandpass filters (see Feature Acquisition, above). Zero crossings where the gradient is oriented vertically are ignored (the implicit camera model has the epipolar lines oriented horizontally). Zero crossings are located to an accuracy of one pixel, and their orientations are recorded in increments of 30°. Matching occurs at different spatial scales in separate "channels," with the matches found at the larger scales establishing a rough correspondence for the smaller scales. In other words, the lower frequency channels are used to narrow the search for matches in the higher frequency channels. Within a channel, matching proceeds as follows. First, a zero crossing is located in one image. The region surrounding the approximate location in the second image is then divided into three pools—two larger positive- and negative-disparity pools and a smaller zero-disparity pool centered on the predicted match location. The three pools together span a region twice the width of the central positive region of the convolution mask. Zero crossings from pools in the second image can match the zero crossing from the first image only if they result from convolutions of the same size mask, have the same contrast direction, and have approximately the same orientation. If a unique match is found (i.e., if only one of the pools has a zero crossing satisfying the above criteria), the match is accepted as valid. If two or three candidate matches are found, they are saved for future disambiguation. After all matches have been found (ambiguous or not), the ambiguous ones are resolved by searching through the neighborhoods of the points to determine the dominant disparity (positive, negative, or zero).

Zero crossings are quite sensitive to the presence of noise. Even small noise levels can cause the zero-crossing geometry to vary dramatically. Nishihara (34) described a variation of the Marr-Poggio approach that is more successful with noisy imagery. Instead of matching explicit zero-crossing points, he matches the sign of the result of convolving the difference-of-Gaussians filter with the intensity images. The sign of the convolution degrades more uniformly with increasing noise than does the coherence of zero-crossing contours.

Kass (35) describes a matching technique in which measurements are derived by applying several linear shift-invariant filters to the two images to produce a feature vector for each pixel. A match is established whenever the distance between two feature vectors falls below some threshold. A sto-

chastic image model is used to derive appropriate design parameters that optimize the performance of the system in terms of false-positive and false-negative error rates and resolution of disparity. He describes an implementation that uses 12 features: The images are smoothed with Gaussian filters that have standard deviations  $\sigma_1$ ,  $\sigma_2$ , and  $\sigma_3$  in the ratio  $\sigma_1/\sigma_2 = \sigma_2/\sigma_3 = 2.5$ ; two first derivatives and two second derivatives of each filtered image comprise the feature vector. Good results were obtained in experiments performed on both random-dot stereograms and vertical aerial images.

**Search Strategies.** A typical stereo task might require two  $512 \times 512$  pixel images to be matched with a disparity range of 30 pixels (assuming the epipolar constraint is used). An exhaustive search (qv) would require almost  $8 \times 10^6$  match decisions, and if global evidence is used to resolve ambiguity, the complexity-of-search problem would be much worse (see Search). Of course, feature-matching techniques do not attempt to match every pixel location, but if small-scale features are used, exhaustive search can still be very costly. For this reason, considerable research in computational stereo has focused on developing effective search strategies.

One global constraint that has been widely employed for search reduction and resolution of ambiguous matches is the continuity constraint: Because the scene consists of mostly continuous surfaces at the level of resolution required for meaningful interpretation, disparity will generally vary continuously except at relatively rare occlusion boundaries. (Prazdny (36) notes that the continuity of depth (or disparity) is a special case of the coherence of physical surfaces. A strict continuity assumption prevents stereo systems from handling such real-world situations as "semitransparent" vegetation, in which a superposition of several interlaced, continuous disparity fields results in locally discontinuous disparities. Prazdny presents a noniterative, parallel procedure that detects disparities generated by both opaque and transparent surfaces.)

Of course, the application of the continuity constraint requires that aggregates of consistent local matches be identified. Marr and Poggio proposed a "cooperative" computational model for human stereo vision that uses the continuity constraint to match random-dot stereograms (9). In this scheme multiple disparity assignments of a point inhibit one another, and local collections of similar disparities support one another. Barnard described a relaxation-labeling method that used the continuity constraint to match point features obtained with the Moravec interest operator (23). The epipolar constraint is not required, so this method can be used for the sparse matching step preliminary to determining a camera model as well as for dense matching.

Coarse-to-fine search strategies have been used extensively for stereo matching. The essential idea is to obtain coarse disparity estimates quickly, which can be used to constrain finer-resolution matching. This process can be repeated in several stages. Already discussed is the way the Marr-Poggio approach uses matches of large-scale features to bring smaller scale matching channels into rough correspondence. Moravec developed a stereo system for autonomous vehicle guidance that used coarse-to-fine search to establish correspondences among a series of stereo images (6,7); Hannah (22,32) uses a similar approach. Quam (37) described a method for constructing digital terrain models from aerial stereo images that uses a coarse-to-fine hierarchical control structure both for constraint propagation and efficiency. In this scheme disparity

estimates from coarser levels are also used to warp the stereo images, effectively removing perspective distortions and improving cross-correlation-based matching at finer levels.

Baker and Binford use a dynamic-programming search strategy (25) in addition to coarse-to-fine matching. Their system matches edges along epipolar lines by finding a set of correspondences that maximizes a measure of "goodness" based on local edge properties. The Viterbi dynamic programming algorithm (qv) (38) is used to find these correspondences efficiently by taking advantage of a partitioning constraint that separates the correspondence problem into independent subproblems; namely, the edges are assumed to occur in the same order in both images (positional reversals do occur in stereo images, however, as the authors note). This correspondence process proceeds independently for each epipolar line (i.e., each scanline in rectified imagery). Correspondences are refined in a subsequent step by deleting those matches that break the continuity of edges across epipolar lines.

**Distance Determination.** With few exceptions, work in image understanding (qv) has not specifically dealt with the problem of distance determination. The matching (qv) problem has been considered the hardest and most significant problem in computational stereo. Once accurate matches have been found, the determination of distance is a relatively simple matter of triangulation (Fig. 2). Nevertheless, this step can present significant difficulties if the matches are somewhat inaccurate or unreliable.

To a first approximation, the error in stereo distance measurements is directly proportional to the positional error of the matches and inversely proportional to the length of the stereo baseline (i.e., the distance between the cameras). Lengthening the stereo baseline complicates the matching problem by increasing both the range of disparity (i.e., the area that must be searched) and the difference in appearance of the features being matched.

In many cases matches are made to an accuracy of only a pixel. However, both the area-correlation and the feature-matching approaches can provide better accuracy. Subpixel accuracy using area correlation requires interpolation over the correlation surface. Although some feature-detection methods can locate features to accuracies better than one pixel, this is largely dependent on the type of operator used, and there are no generally applicable techniques.

Another approach is to settle for one-pixel accuracy but to use multiple views (6). In Moravec's scheme a match from a particular pair of views represents a depth estimate whose uncertainty depends on both the accuracy of the match and on the length of the stereo baseline. Matches from many pairs of views can be averaged to obtain a more accurate estimate. The contribution of a match to the final-depth estimate can be weighted according to any of the factors that bear on the confidence or accuracy of the match.

A special problem of considerable importance (e.g., in computer-aided design) is that of constructing a three-dimensional model from a pair of line drawings. Given that the line drawings are in correspondence and that the objects they represent are polyhedral solids, the main problems to be addressed are

1. disambiguating matches between vertices when some of the vertices are superimposed on each other in one or both of the line drawings (a common occurrence in engineering



drawings) or when the back projections of the lines and vertices have more than one intersection and

2. determining which volumetric extents of the "wire frame" determined in 1 are empty and which are solid.

Algorithms for these purposes are presented by Wesley and Markowsky (39) and Strat (40).

**Interpolation.** Stereo applications usually demand a dense array of depth estimates that the feature-matching approach cannot provide because "reliable" features are sparsely and irregularly distributed over the images. Correlation matching is more suited than feature matching to obtaining dense matches, although it tends to fail in areas for which it has low information. Consequently, either approach usually requires some kind of interpolation step.

The most straightforward way to derive a dense depth array from a sparse one is simply to treat the sparse array as a sampling of a continuous depth function and to approximate the continuous function using a conventional interpolation or approximation method. If the sparse depth array is complete enough to capture the important changes in depth, this approach may be adequate. Aerial stereophotographs of rolling terrain, e.g., can be successfully handled in this way. Smith (41) describes a multiquadratic surface-fitting technique that is well-suited to modeling natural terrain from aerial stereophotography. Grimson (11) has noted that the absence of matchable features implies a limit on the variability of the surface to be interpolated between features and has described a spline interpolation procedure based on this observation equivalent to the "thin-plate" physical model of elasticity theory. Terzopoulos (42) has developed a fast multilevel implementation of this idea using the method of finite elements. Oblique views of complex scenes will usually contain occlusion edges, and surface-fitting techniques that require continuity or smoothness everywhere will not be appropriate. Terzopoulos has proposed a generalization of the thin-plate spline interpolation method that uses lower order splines to accommodate discontinuities and lack of smoothness (43).

Another approach to the interpolation problem is to fit a priori geometric models to the sparse depth array. Normally, model fitting would be preceded by clustering to find the subsets of points in three-dimensional space corresponding to significant structures in the scene. Each cluster would then be fit to the best available model, thereby instantiating the model's free variables and providing an interpolation function. This approach has been used to find ground planes (24), elliptical structures in stereophotographs (8), and smooth surfaces in range data acquired with a laser rangefinder (44).

## Discussion

Automated computational stereo cannot simply duplicate the steps and procedures currently employed when a human interpreter is an integral part of the process. There is at present no reasonable way to duplicate the human ability to invoke semantic and physical knowledge to filter out gross errors in the various steps, especially in those steps involved in matching. Techniques that are highly tolerant of errors [such as RAN-SAC (21)] will have to be substituted for those that require reliable manually filtered data (such as least-squares estima-

tion of camera parameters). Constraints indirectly invoked by the human interpreter must be made explicit and embedded directly into the automated procedures (e.g., the fact that all vertical edges depicted in an image must pass through a common vanishing point). Automated stereo, not limited by the two-image constraint of human vision, can partially compensate for the lack of a human knowledge base by "simultaneously" processing a large number of views of a scene to resolve ambiguity and by approaching some of the problems from a quantitative (model-based) approach rather than the qualitative (constraint-based) approach of humans.

Most of the approaches to stereo compilation involve matching localized features or intensity patches to determine corresponding points in the two images, solving simultaneous equations to obtain the parameters of the camera model, and using trigonometry to obtain scene depths. This simple paradigm fails when the views are dissimilar (e.g., due to perspective distortion of nearby objects, unequal characteristics of the two sensors, or highly directional illumination effects such as specularities) or are really ambiguous (e.g., due to featureless areas, repeated structure such as plowed fields or windows on the face of a building) or contain occluded areas so that portions of the scene are not visible in one or the other of the two images. This collection of problems, although rare in aerial mapping tasks, are pervasive for a robot moving through a complex, natural three-dimensional environment. The human visual system routinely solves these problems in ways that are still not understood. Certainly global context and semantic knowledge play a role, but existing technology has nothing comparable to offer. It is probably also the case that, rather than attempting to obtain a uniformly dense array of scene depths, the goal-oriented human visual system directly extracts more organized structures from the scene (e.g., coherent objects) and is efficient in deciding where detailed examination is required. The question of scene representation beyond an array of depths is still completely open. Although one can automate much of the process of geometric scene modeling from an aerial perspective (as required for cartographic applications) and can deal with constrained viewing conditions common in industrial automation tasks, an adequate approach has not yet been devised for the problem of modeling outdoor three-dimensional scenes. It appears that a solution to this problem will require the use of knowledge and reasoning, rather than simple direct matching of the immediately available image data.

## BIBLIOGRAPHY

1. C. Konecny and D. Pape, "Correlation techniques and devices," *Photogram. Eng. Remot. Sens.* **47**(3), 323-333 (March 1981).
2. J. B. Case, "Automation in photogrammetry," *Photogram. Eng. Remot. Sens.* **47**(3), 335-341 (March 1981).
3. L. W. Fritz, "Automated correlation of stereo data," *Photogram. Eng. Remot. Sens.* **49**(4), 527-544 (April 1983).
4. M. M. Thompson (ed.), *Manual of Photogrammetry*, 3rd ed., American Society of Photogrammetry, Falls Church, VA, 1966.
5. M. J. Hannah, Bootstrap Stereo, *Proceedings of the Image Understanding Workshop*, College Park, MD, April 1980, pp. 201-208.
6. H. Moravec, Visual mapping by a robot rover, *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan, August 1979, pp. 598-600.

7. H. Moravec, Rover Visual Obstacle Avoidance, *Proceedings of the Seventh International Joint Conference Artificial Intelligence* Vancouver, British Columbia, August 1981, pp. 785–790.
8. D. B. Gennery, Object detection and measurement using stereo vision, *Proceedings of the Image Understanding Workshop*, College Park, MD, April 1980, pp. 161–167.
9. D. Marr and T. Poggio, "Cooperative computation of stereo disparity," *Science* **194**, 283–287 (1976).
10. W. E. L. Grimson, "A computer implementation of a theory of human stereo vision," *Philos. Trans. Roy. Soc. Lond. Ser. B* **292**, 217–253 (1981).
11. W. E. L. Grimson, *From Images to Surfaces: A Computational Study of the Human Early Visual System*, MIT Press, Cambridge, MA, 1981.
12. B. Julesz, *Foundations of Cyclopean Perception*, the University of Chicago Press, Chicago, IL, 1971.
13. D. Marr and T. Poggio, A theory of human stereo vision, Memo 451, Artificial Intelligence Lab, MIT, Cambridge, MA, November 1977.
14. D. Gennery, Stereo-camera calibration, *Proceedings of the Image Understanding Workshop*, Los Angeles, CA, November 1979, pp. 101–107.
15. F. H. Thompson, *Photogram. Rec.* **3**(14), 152–159 (1959).
16. H. C. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature* **293**, 133–135 (September 10, 1981).
17. R. Y. Tsai and T. S. Huang, "Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-6**(1), 13–27 (January 1984).
18. T. M. Strat and M. A. Fischler, One-eyed stereo: a general approach to modeling 3-D scene geometry, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985, pp. 937–943.
19. S. Ganapathy, Decomposition of transformation matrices for robot vision, IEEE Reference: CH2008-1/84/000/0130, 1984, pp. 101–107.
20. T. M. Strat, Recovering the camera parameters from a transformation matrix, *Proceedings of the Image Understanding Workshop*, New Orleans, LA, October 1984, pp. 264–271.
21. M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *CACM* **24**(6), 381–395 (June 1981).
22. M. J. Hannah, Computer Matching of Areas in Stereo Imagery, Ph.D. dissertation, AIM 239, Computer Science Department, Stanford University, Stanford, CA, 1974.
23. S. T. Barnard and W. B. Thompson, "Disparity analysis of images," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-2**(4), 333–340 (July 1980).
24. D. Arnold, Local Context in Matching Edges for Stereo Vision, *Proceedings of the Image Understanding Workshop*, May 1978, pp. 65–72.
25. H. Baker and T. O. Binford, Depth from Edge and Intensity Based Stereo, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, British Columbia, August 1981, pp. 631–636.
26. J. E. W. Mayhew and J. P. Frisby, "Psychological and computational studies towards a theory of human stereopsis," *Artif. Intell.* **17**, 349–385 (August 1981).
27. B. F. Logan, "Information in the zerocrossing of bandpass signals," *Bell Sys. Technol. J.* **56**, 487–510 (1977).
28. S. Ganapathy, Reconstruction of scenes containing polyhedra from stereo pairs of views, Memo AIM-272, Artificial Intelligence Lab, Stanford University, December 1975.
29. M. Herman and T. Kanade, The 3D MOSAIC scene understanding system: incremental reconstruction of 3-D scenes from complex images, *Proceedings of the Image Understanding Workshop*, New Orleans, LA, October 1984, pp. 137–148.
30. D. J. Panton, "A flexible approach to digital stereo mapping," *Photogram. Eng. Remot. Sens.* **44**(12), 1499–1512 (December 1978).
31. S. T. Barnard, A stochastic approach to stereo vision, Technical Note 373, Artificial Intelligence Center SRI International, Menlo Park, CA, March 1986; also in Proc. of AAAI, Philadelphia, PA, August 1986, pp. 676–680.
32. M. J. Hannah, SRI's baseline stereo system, *Proceedings of the Image Understanding Workshop*, Miami Beach, FL, December 1985, pp. 149–153.
33. E. C. Hildreth, *The Measurement of Visual Motion*, MIT, Cambridge, MA, 1984.
34. H. K. Nishihara, "Practical real-time imaging stereo matcher," *Opt. Eng.* **23**(5), 536–545 (September/October 1984).
35. M. Kass, Computing Visual Correspondences, in *Proceedings of the Image Understanding Workshop*, Arlington, VA, June 1983, pp. 54–60.
36. K. Prazdny, "Detection of binocular disparities," *Biol. Cybernet.* **52**, 93–99 (1985).
37. L. Quam, Hierarchical warp stereo, *Proceedings of the Image Understanding Workshop*, New Orleans, LA, October 1984, pp. 149–155.
38. G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE* **61**(3), 268–278 (March 1973).
39. M. A. Wesley and G. Markowsky, "Fleshing out projections," *IBM J. Res. Devel.* **25**(6), 934–954 (November 1981).
40. T. M. Strat, Spatial reasoning from line drawings of polyhedra, *Proceedings of the Workshop on Computer Vision: Representation and Control*, Annapolis, MD, April 1984, pp. 219–224.
41. G. B. Smith, A fast surface interpolation technique, *Proceedings of the Image Understanding Workshop*, New Orleans, LA, October 1984, pp. 211–215.
42. D. Terzopoulos, "Multilevel computational processes for visual surface reconstruction," *Comput. Graph. Img. Proc.* **24**(1), 52–96 (October 1983).
43. D. Terzopoulos, Controlled-smoothness stabilizers for the regularization of ill-posed visual problems involving discontinuities, *Proceedings of the Image Understanding Workshop*, New Orleans, LA, October 1984, pp. 225–229.
44. R. O. Duda, D. Nitzan, and P. Barrett, "Use of range and reflectance data to find planar surface regions," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-1**(3), 259–271 (July 1979).

STEPHAN T. BARNARD AND MARTIN A. FISCHLER  
SRI International

Support for the preparation of an early version of this entry was provided under DARPA Contract No. MDA903-79-C-0588. We thank the ACM for permission to use material from the original version, which was published under the title "Computational stereo," by S. T. Barnard and M. A. Fischler, *Comput. Surv.* **14**(4), 553–572 (December 1982).

## STORY ANALYSIS

### Understanding and Inference

Computational models for narrative text comprehension comprise one of the most ambitious undertakings in natural-lan-

guage-processing research. Before describing any specific story-understanding systems, it is appropriate to first discuss the goals and underlying premises of this research effort. Just what does it mean to understand a story? How is it possible to determine if a computer program was successful in this undertaking? For that matter, how is it possible to determine when people are successful in understanding a story?

In order to evaluate computer programs that claim to "understand" text, it is necessary to ask how reading comprehension in people is evaluated. When deciding whether a person has understood a story, there are only a few tests available: One can ask questions about the story, one can ask for a paraphrase of the story, or one can ask for a summary of the story. These are the only criteria available for evaluating computer programs as well. It follows that anyone designing a story-understanding system is wise to construct additional capabilities to demonstrate that system's comprehension in terms of at least one of these three possible criteria. Even the most rigorously designed system will be impossible to evaluate if one is forced to examine internal memory representations in the absence of retrieval mechanisms that utilize these representations.

All story-understanding systems must construct some memory representation (see Representation, knowledge) for the story being read. In many ways understanding is tantamount to remembering. When people read stories, they do not remember the precise wording of the original text. What they remember is the conceptual content of the story—characters, event sequences, problems, and goal-oriented behavior. Since there is no precise psychological theory of exactly what conceptual content is retained, researchers designing text-understanding systems must constantly fall back on question answering (qv), paraphrase, and summarization in order to evaluate their progress. If people can remember the name of the main character, it is appropriate for a computer simulation to remember that name as well. If people tend to forget the names of minor characters, it may be acceptable for a computer program to fail in recalling these names as well.

Although it is difficult to say precisely what most people would remember in response to any specific story, it is possible to characterize human text comprehension in more general terms. When people read stories, they generate a large number of inferences (qv) in order to make sense of the explicit story text. An inference is an assumption that could be wrong, and it is the ability to make reasonable assumptions that makes narrative text comprehension a difficult problem for computers. If a story could be written to make explicit all of the information that would otherwise be inferred, it would be incredibly long, tedious to read, and infuriatingly detailed.

The problem of inference was first tackled by Rieger (1) as a problem of causal chain completion. Suppose you were told:

John hit Mary.

Mary was taken to the hospital.

You would infer that John probably hit Mary with his hand (perhaps a fist), that Mary was physically hurt as a result of John hitting her, that someone (probably not John) became aware of Mary's condition, and that person took Mary to the emergency room of the nearest hospital. Mary might have been taken by an ambulance or she might have been driven in a car, but she probably did not get on a plane or an ocean liner.

These are all assumptions that could be wrong. Yet if someone reading these two sentences did not make these assumptions, it would be doubtful if he/she truly understood the two sentences.

Processes of inference are the key to narrative text comprehension. It has been estimated that for every concept explicitly spelled out in a narrative, there are eight more that must be generated by inference (2). If a system (human or otherwise) fails to generate these inferences, a functional level of comprehension has not been attained.

In Rieger's model of inference consecutive sentences were first parsed into conceptual-dependency (qv) representations. These representations were then used as the starting point for an intersection search (see Search, bidirectional) within expanding pools of inference. Rieger hypothesized 16 classes of inference along with mechanisms that would generate these plausible inferences in response to conceptual-dependency representations. If an intersection could be found within the two pools of first-generation inferences, a simple causal chain could be constructed between the two original input concepts. For example, using the two sentences above, one can derive:

John hit Mary → John wanted to hurt Mary  
 John hit Mary with his hand  
 John was physically close to Mary  
 Mary was hurt

Mary was taken to the hospital →  
 Mary had been hurt  
 Someone took Mary to the hospital  
 A hospital was nearby

Since there is an intersection in the first generation, it is possible to construct a causal chain connecting the two input concepts:

John hit Mary → Mary was hurt →  
 Mary was taken to the hospital

If no intersecting concept had been found, it would have been necessary to recursively expand additional inferences out from each of the first generation concepts and search for an intersection within this larger inference pool. Recursive expansions of the inference space would continue until a connection was found.

Rieger was using a classical technique called spreading activation in his model for causal chain completion. Unfortunately, his model shared the usual problems encountered with spreading activation: the search space grew at an exponential rate, making it very costly to build paths of any length, and the model could not identify causally incoherent input pairs. Given two totally unrelated concepts, Rieger's model would continue to search for a connecting chain indefinitely. There was no way to tell the system when to give up.

Although Rieger's work on inference was flawed, it nevertheless inspired researchers to address the problems of inference underlying text comprehension. Rieger's model was too unconstrained and willing to examine too many possible paths, but how could inferences be generated in a more directed and efficient manner? Many researchers have examined a number of alternatives since Rieger's pioneering effort, and all subsequent models of inference share one common premise: The solution involves large amounts of very specific knowledge about the world. Commonsense reasoning (qv) cannot be

achieved with general heuristics (qv) alone; large amounts of general world knowledge are needed as well.

### Using World Knowledge: Scripts

Once one is committed to making large amounts of knowledge available to a text understander, one must ask how this information can be organized and accessed in an effective and efficient manner and what sort of knowledge is needed? Would it suffice to access an on-line encyclopedia of world knowledge? Exactly what is being talked about when one says there is a need for general world knowledge?

The computer programs described below were all designed in an effort to answer these questions about knowledge-based text comprehension. The knowledge needed is not the kind found in encyclopedias. Story-understanding systems constructed by AI researchers involve very different sorts of knowledge about the world, and the problem of identifying this knowledge amounts to a sort of *epistemology* (qv). The problems of narrative text comprehension are primarily problems in knowledge representation.

The first computer program to read stories and answer questions about those stories was the SAM (Script Applier Mechanism) system (3,4). SAM (qv) relied on four major processing modules: a natural-language sentence analyzer, a script-application mechanism, a natural-language sentence generator, and a question-answering module. The sentence analyzer and sentence generator were based on computer programs that were originally designed for the MARGIE system by Riesbeck and Goldman (5). The heart of SAM was the script applier, a knowledge-based inference mechanism based on theories first proposed by Schank and Abelson (6). The question-answering module used to demonstrate SAM's comprehension was designed by Lehnert (7).

In order to understand how SAM worked, it is necessary to first understand the idea of script-based comprehension. A script (qv) is a specific type of knowledge structure used to generate inferences about stereotypic event sequences. Much of the knowledge about the world can be described in terms of stereotypic event sequences. There is a sequence of events for eating at a restaurant, for getting gas at a gas station, for buying a car, for seeing a doctor, and so on. The list of things that can be described in terms of event sequences is quite large.

For example, the event sequence associated with going to a movie involves getting to the movie theater, standing in line for tickets, buying the tickets, going inside the theater, handing your ticket to an usher, taking the ticket stub back from the usher, finding a seat inside the theater, sitting down, waiting for the movie to begin, sitting through the movie, leaving the theater, and going home again. This path of events describes a "default" trip through the movie script. In addition to this default path there are some optional side trips that involve buying candy inside the theater, going to the washroom, or possibly playing a video game before the show starts. These optional paths within the movie script are not always possible or always exercised, but they are common enough to be included in one's knowledge structure about movie theaters. The movie script is designed to include all of the standard knowledge an American is likely to have about the events involved in going to a movie.

The role of scripts in natural-language comprehension is

critical (8). If I say "I went to a movie last night," it is important to understand this as more than the literal event of physical movement to a movie theater. Any human who understands this sentence will make a number of inferences about exactly what I was doing last night. I must have had money to buy tickets, I spent a couple of hours watching a movie inside a theater, I may have eaten some junk food, I may still have my ticket stub, etc. All of this information can be relevant to a larger conversational context in any number of ways. My students may be asking me why their exams did not get graded last night, or a relative may want to know why I did not answer the telephone. If I explain that I went to a movie last night, my interrogators should be capable of seeing how that simple statement answers their question by addressing resource limitations on my time or simple enabling conditions with respect to my physical location.

People are remarkably adept at generating appropriate inferences and using them to establish causal connections in order to explain human behavior. A script is just one knowledge structure designed to make inferences available. Scripts do not account for all of the inferences a language understander needs, but they do account for a healthy subset of natural-language inferences. In the context of story comprehension, the term "script application" refers to an inference process that generates coherent causal chains by filling in gaps of missing, but implicit information. If two speakers do not share the same scriptal knowledge, communication will suffer accordingly. For example, if I tell someone who knows nothing about computers that I had to reboot my personal computer yesterday, I should not expect that person to make any reasonable assumptions about how much time it took or what was involved. If that information is going to be important, I had better explain my actions more explicitly.

The SAM system operated with a knowledge base of scripts that enabled SAM to make inferences about events that were not explicitly described in the input story. For example, if SAM was told that John went to a restaurant and had lobster, SAM would infer that John went inside the restaurant, sat down at a table, ordered lobster, and was served. If the text went on to say that John went to a movie afterward, SAM would infer that John ate the lobster, received a check, paid the check, and left the restaurant. All of these inferences would be made on the basis of SAM's scriptal knowledge about stereotypical events in a restaurant.

Using a script for car accidents, SAM could understand simple newspaper stories. The following newspaper account was actual input for SAM:

*Friday evening a car swerved off Route 69. The vehicle struck a tree. The passenger, a New Jersey man, was killed. David Hall, 27, was pronounced dead at the scene by Dr. Dana Blanchard, medical examiner. Frank Miller, 32, of 593 Foxon Rd., the driver, was taken to Milford Hospital by Flanagan Ambulance. He was treated and released. The passenger was extricated from the vehicle by the Branford Fire Department. No charges were made. Patrolman Robert Onofrio investigated the accident.*

Having generated a memory representation for this story, SAM can answer questions and paraphrase the source text:

Was anyone killed?

YES, DAVID HALL DIED.

Was anyone hurt?  
YES, FRANK MILLER WAS SLIGHTLY INJURED.

Why was Frank Miller hurt?  
BECAUSE THE AUTOMOBILE HIT A TREE.

Did Miller go to the hospital?  
YES, HE WAS IN THE MILFORD HOSPITAL.

How did Miller get to the hospital?  
THE FLANAGAN AMBULANCE COMPANY TOOK HIM  
TO THE MILFORD HOSPITAL.

SAM's summary:

AN AUTOMOBILE HIT A TREE NEAR HIGHWAY 69  
FOUR DAYS AGO. DAVID HALL, AGE 27, RESIDENCE  
IN NEW JERSEY, THE PASSENGER, DIED. FRANK  
MILLER, AGE 34, RESIDENCE AT 593 FOXON ROAD IN  
NEW HAVEN CONNECTICUT, THE DRIVER, WAS  
SLIGHTLY INJURED. THE POLICE DEPARTMENT DID  
NOT FILE CHARGES.

Each input sentence to SAM was first parsed into a conceptual representation using the representational system of conceptual dependency (5). The process of script application was then called to locate a matching event within whatever script or scripts were currently active. Once a match was made, SAM instantiated a chain of events drawn from the scriptal structure leading up to the input event. Eventually, a full path through the script would be pieced together into a memory representation for the story being processed (see Fig. 1).

The process of script application did encounter some interesting problems when more than one script had to be active at the same time or when multiple paths through a script were possible. For example, if you hear that John ordered a hamburger at a restaurant, you could reasonably infer that John was served a hamburger, that John ate the hamburger, that John paid a check, and that John then left the restaurant. But if you hear that John ordered a hamburger, waited for 45 minutes, and left, your inferences will be very different. In this case John was most likely not served, he probably did not eat, and he undoubtedly did not pay. In addition to the default path through the restaurant script, there are alternative paths to take when something goes wrong. In this case a quick exit from the restaurant script was taken before the serving scene could take place.

It was necessary to access the scriptal knowledge base at the time of question answering (qv) as well as at the time of

story understanding: Some questions could not be answered on the basis of the story's memory representation alone (7). This suggests that answering questions about stories is somewhat more complicated than straightforward database retrieval: Specific heuristics and long-term knowledge structures are needed in addition to whatever memory representation has been generated for the story at hand.

This early work on knowledge-based story analysis captured the interest of many cognitive psychologists (see Cognitive psychology) who subsequently designed experiments to test the validity of script application as a model of human information processing (9,10). Although it is very difficult to design experiments for such complex cognition, much of the data obtained in these experiments has inspired further empirical research in knowledge-based information processing.

The notion of script-based understanding proved useful for a number of other computer programs after SAM. DeJong implemented the FRUMP (qv) system (Fast Reading Understanding and Memory Program) to see how scripts would enable a system to "skim" an input text (11). FRUMP read stories from a UPI wire using "sketchy scripts" that were simplified script structures designed to aid in sentence analysis as well as memory instantiation. Here is an example of FRUMP processing a UPI story:

INPUT:

MOUNT VERNON, IL (UPI)—A SMALL  
EARTHQUAKE SHOOK SEVERAL SOUTHERN ILLINOIS  
COUNTIES MONDAY NIGHT, THE NATIONAL  
EARTHQUAKE INFORMATION SERVICE IN GOLDEN,  
CO, REPORTED.

SPOKESMAN DIN FINLEY SAID THE QUAKE  
MEASURED 3.2 ON THE RICHTER SCALE, "PROBABLY  
NOT ENOUGH TO DO ANY DAMAGE OR CAUSE ANY  
INJURIES." THE QUAKE OCCURRED ABOUT 7:48 PM  
CST AND WAS CENTERED ABOUT 30 MILES EAST OF  
MOUNT VERNON, FINLEY SAID. IT WAS FELT IN  
RICHLAND, CLAY, JASPER, EFFINGTON AND MARION  
COUNTIES.

SMALL EARTHQUAKES ARE COMMON IN THE  
AREA, FINLEY SAID.

SELECTED SKETCHY SCRIPT \$EARTHQUAKE

CPU TIME FOR UNDERSTANDING = 3040  
MILLISECONDS

ENGLISH SUMMARY:

THERE WAS AN EARTHQUAKE IN ILLINOIS WITH 3.2  
RICHTER SCALE READING.

DeJong estimated that about 50% of the new stories found on the UPI wire were scriptal in nature. Of these, about 10% could be analyzed correctly by FRUMP using 48 scripts and a vocabulary of about 1200 words. In general, the most difficult problem for FRUMP was determining which script should be activated for a given story. Once the proper script was called for an input story, the rest was relatively easy.

#### Using World Knowledge: Goals and Plans

At about the same time that DeJong was working on sketchy scripts, Carbonell was implementing a system called POLI-

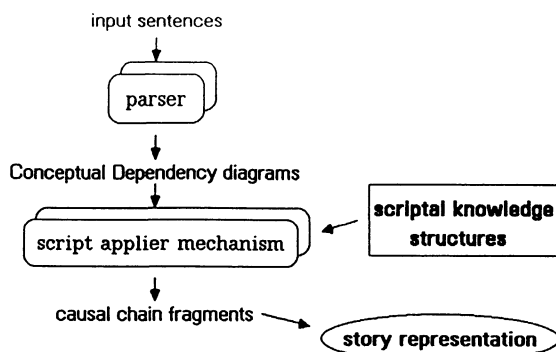


Figure 1. SAM during story comprehension.

TICS (qv) that analyzed world events from the perspective of specific ideologies or political belief systems (12). Given an event and a political ideology, POLITICS would interpret the event by generating a full story representation, predicting possible future events, answering questions, and suggesting how the United States should respond. The following dialogue is an actual interactive session with POLITICS illustrating its analysis of a single event from two ideological vantage points:

U.S.—Conservative Interpretation of Event (1) (9):

INPUT TEXT: The United States Congress voted to fund the Trident submarine project.

Q: What did the U.S. Congress do this for?

A: THE UNITED STATES CONGRESS WANTS THE UNITED STATES ARMED FORCES TO BE STRONGER.

Q: Why should the U.S. be stronger?

A: THE UNITED STATES ARMED FORCES SHOULD BE STRONG TO STOP COMMUNIST EXPANSION.

Q: What should the U.S. do if Russia builds nuclear submarines?

A: THE UNITED STATES SHOULD BUILD MORE SUBMARINES.

U.S.—Liberal Interpretation of Event (1):

INPUT TEXT: The United States Congress voted to fund the Trident submarine project.

Q: What did the U.S. Congress do this for?

A: THE UNITED STATES CONGRESS FEARED FALLING BEHIND IN THE ARMS RACE.

Q: What will Russia do next?

A: RUSSIA WILL PROBABLY BUILD MORE WEAPONS.

Q: What should the U.S. do if Russia builds submarines?

A: THE UNITED STATES SHOULD NEGOTIATE WITH RUSSIA TO STOP THE ARMS RACE.

POLITICS was based in part on scriptal knowledge structures: Scripts were used to organize knowledge about governmental activities and world events like government funding of large projects and the arms race. But POLITICS went beyond scriptal knowledge structures to employ knowledge about goals, goal conflicts, plans, and counterplans as well. For example, political ideologies were encoded as goal trees. Using this approach, POLITICS was able to successfully model both conservative and liberal ideologies associated with U.S. foreign policy. The highest nodes in the goal tree for the Soviet Union under a conservative ideology are shown in Figure 2.

POLITICS was an important experiment in story comprehension because it illustrated the importance of multiple knowledge structures, particularly the use of goal-oriented knowledge in addition to stereotypical event sequences.

Important work with goal-based understanding was also being pursued by Wilensky in the development of the PAM (Plan Applier Mechanism) system (13,14). PAM (qv) was de-

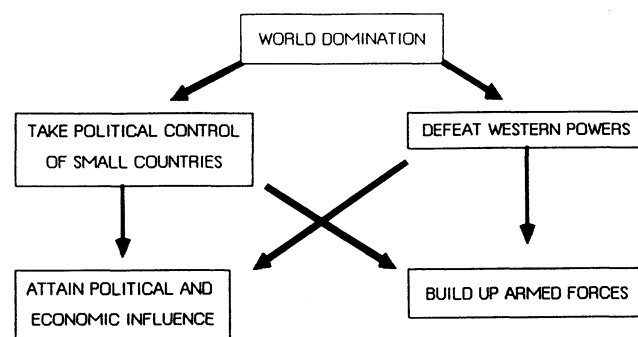


Figure 2. Soviet goal tree (partial).

signed to exploit knowledge about human goals and plan-oriented behavior. Life would be rather dull if every situation encountered could be handled on the basis of stereotypical event sequences. Various situations periodically arise that demand creative problem solving, and PAM was devised to show how a story-understanding system could organize inferences about human problem solving.

Using a taxonomy of goals developed at Yale University (6), PAM generates inferences about human behavior that can be explained in terms of goals and plans for realizing those goals. Like SAM, FRUMP, and POLITICS, PAM is a predictive story-understanding system, but PAM's predictions are more abstract than the predictions generated by script application. Although script-based understanders can make predictions about specific events, PAM must make general predictions about certain classes of behavior. For example, suppose PAM were given as input:

1. John wanted to be elected selectman.

Although a seasoned politician might have some scriptal knowledge structures associated with winning elections, most people would not have specific event sequences stored in their memory to access in this situation. Yet people still have knowledge about elections and what a candidate must generally do in order to win one. It is possible to find satisfactory causal connections between sentence 1 and each of the following sentences:

2. He asked the mayor for his support.
3. He bought a new suit.
4. He hired someone to print up campaign signs.
5. He read books by successful politicians.

At the same time, there are plenty of sentences that fail to make sense in the context of 1:

6. He asked his veterinarian for flea powder.
7. He bought a new set of garden tools.
8. He hired someone to pave his driveway.
9. He read a lot of science fiction.

In order to establish explanatory connections between 1 and a subsequent sentence, it is necessary to make predictions about general behavior that can help John attain his goal of winning the election. All goal-oriented behavior can be categorized in simple ways. For example, if you want to achieve X,



one possible plan of action is to get someone else to achieve X for you. PAM knows about a general plan called "invoke agency," which can be used to explain sentence 2. In general, someone is supportive if he/she is willing to act as your agent when a situation calls for it. So asking someone for support is a straightforward way to invoke agency. No one would confuse 2 with 6 since 6 could not possibly be interpreted as a request for anything other than flea powder.

In order to understand sentence 3, an intermediate inference must be made about how one wins elections. Elections are won with votes and votes are won with persuasion. The voting public must be persuaded to vote for John. A great deal of interactive goal-oriented behavior involves persuasion in one form or another. If PAM activates its knowledge about persuasion after hearing 1, PAM can generally expect John to engage in a number of possible persuasive plans: ask, invoke interpersonal theme, bargain/favor, bargain/object, inform/reason, threaten, overpower. Buying a new suit makes sense in the context of inform/reason since it tacitly tells the voters that John is a civilized man who respects convention and honors his audience with a reassuring appearance. In effect, John is saying, "You should elect me because I can be trusted to respect social mores." John would be wise to execute some additional plans for persuading his constituency, but a new suit is certainly one reasonable instrument for implicit persuasion. It is not possible to interpret sentence 7, the purchase of garden tools, in the same way.

In a similar manner, having campaign signs (sentence 6) is a good way to tell people you're running for office—an important first step toward persuasion. But paving a driveway (sentence 8) is not instrumental to public persuasion, and finally, reading books by successful politicians (sentence 5) makes sense as a transfer of useful knowledge, whereas science fiction (sentence 9) is far less likely to be of immediate use.

As these examples indicate, general world knowledge is critical for distinguishing obvious causalities from tenuous ones. In order to understand how John's behavior makes sense or fails to make sense, it is necessary to know about mayors, flea powder, new suits, garden tools, campaign signs, gardens, driveways, books, politicians, and science fiction. In fact, it would be very difficult to circumscribe a subset of knowledge that could pop up in the context of John's desire to be elected. Given further explanation, any of sentences 6–9 could be connected to 1. However, a casual reader could not be expected to understand what the connection might be in the absence of further information.

Wilensky investigated a number of issues in understanding plan-driven behavior, including goal hierarchies, goal competition, goal conflict, counterplanning, metaplans, and goal subsumption. These general concepts associated with goal-oriented behavior are frequently present in stories that involve people in everyday situations. Although space limitations prevent description of these ideas in any detail, some brief examples can be provided to illustrate them.

John wanted to watch television but Mary wanted to go out to a movie. John told Mary their car should not be driven before replacing the shocks.

When two people are competing for limited resources, their behavior may be explained in terms of both plans and counterplans. In this case it is first necessary to understand that time limitations probably prohibit John and Mary from achieving

both goals of watching television and seeing a movie, so their goals are in competition with one another. Convincing Mary about the car's shocks is not directly instrumental to John's goal of watching TV, but it is a useful counterplan in response to her (implicit) plan of driving to a movie. (Note that if John and Mary live in Manhattan and normally take a taxi when they go out, John's counterplan of persuasion will not have much of an effect on Mary.) And finally, one must understand that John's argument may indeed convince Mary to abandon her goal since a car with faulty shock absorbers may be dangerous to drive, and Mary presumably has another goal concerned with preserving her life and good health. By consulting a standard goal hierarchy that allows one to compare preservation goals with entertainment goals, one can understand why Mary should allow her preservation goal to override her desire for entertainment when her two goals are placed in conflict. All of these inferences are necessary for a satisfactory analysis of these two sentences.

Consider another simple scenario:

John had been renting a car whenever he needed to drive somewhere, but he finally decided to buy one of his own. He picked up the phone book.

Here is a case of goal subsumption: John's recurring goal of needing to get to various places has forced him to find a better solution to his problem than his old solution. Owning his own car may be preferable to renting for any number of reasons: it may be cheaper, less time consuming, or generally less stressful to John. But whatever his reasons, one can recognize John's behavior in terms of a general plan known as goal subsumption. It is often the case that a recurring goal is best handled by acquiring an instrumental object or establishing a useful relationship. In this case, a car is instrumental to moving about, and John will subsume his transportation goals by controlling his own car. Once one understands that John is engaged in goal subsumption, one can focus on his subgoal of obtaining a car. Now one invokes a metaplan for obtaining physical objects. In general, one obtains an object by (a) locating the object, (b) going to the object, and (c) establishing control over the object. This is a very general sequence of subgoals that can be invoked whenever it is necessary to acquire something. Since one knows that John wants to acquire a car, one can now understand his consulting the phone book as a way of satisfying (a). One infers that John is going to look in the phone book for a car dealer, and this is his first step toward satisfying the goal of getting a car.

Of course, inferences are just assumptions that could be wrong. If the story went on to say that John called his brother Bill to see if Bill would talk him out of it, then the understanding about John's use of the phone book would be revised, and one would further understand that John has not begun to execute any plans directed toward the acquisition of a car. In fact, the status of John's acquisition goal would have to be revised from a certain goal to an uncertain goal.

Although script-based story understanders generate memory representations of causal chains and event descriptions, goal-based understanders use memory representations based on goal-fate graphs and plan-box instantiations. Events are not merely recorded; they must also be explained. The following flow of control (Fig. 3) was used in PAM to find satisfactory explanations for events:

Using a memory representation generated at the time of

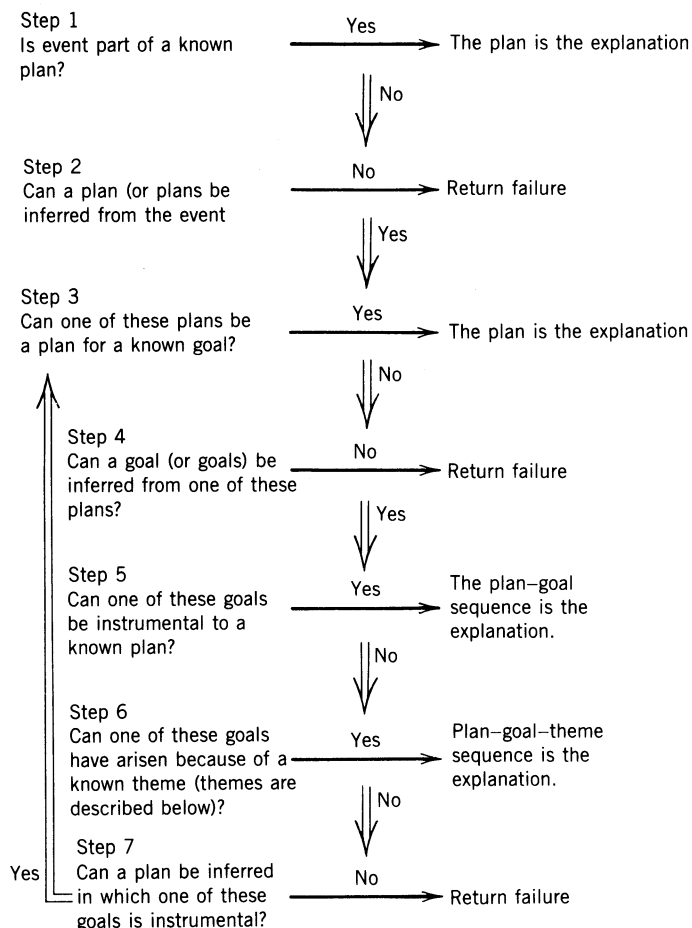


Figure 3. Process of finding an explanation for an event.

understanding, PAM is able to process goal-oriented stories and answer questions about those stories. The following example from Ref. 13 shows actual I/O behavior from PAM:

#### INPUT TEXT:

John wanted money.  
He got a gun and walked to a liquor store.  
He told the owner he wanted some money.  
The owner gave John the money and John left.

INPUT: Why did John get a gun?

OUTPUT: Because John wanted to rob the liquor store.

INPUT: Why did John threaten the shopkeeper?

OUTPUT: Because John needed to get some money.

INPUT: Why did the shopkeeper give John the money?

OUTPUT: Because the shopkeeper didn't want to get hurt.

INPUT: Why did John leave?

OUTPUT: Because John didn't want to get caught.

#### Acquiring World Knowledge

All of the story-understanding systems described so far have been designed to produce memory representations for input stories by accessing fixed knowledge structures in order to generate necessary inferences. A different but related problem

addresses the question of how these long-term knowledge structures are created in the first place. In SAM, FRUMP, POLITICS, and PAM, all of the knowledge structures used were "hand-coded" by the system designer. Lebowitz (15) decided to address the problem of how input stories might alter long-term memory structures: He wanted to investigate how a system could learn from what it reads.

The problem of learning knowledge structures from scratch is very difficult (see Learning), so Lebowitz decided to start with a slightly easier subproblem. Given an existing knowledge structure, how can input stories alter that structure through processes of generalization? By hypothesizing a model for concept generalization, it would be possible to hand-code a simple initial knowledge structure, and then let the story understander refine and extend this knowledge based on individual instances encountered in input stories. The computer program implemented to investigate this problem was called IPP (Integrated Partial Parser), and it operated in the domain of political terrorist stories (15).

Given some basic knowledge about international terrorist attacks to begin with, IPP can process stories describing terrorism and hypothesize various generalizations in response to what it reads:

UPI, 4 April 80, Northern Ireland

Terrorists believed to be from the Irish Republican Army murdered a part-time policeman and firebombed a downtown cafe forcing Good Friday worshippers to evacuate a nearby Protestant Cathedral.

UPI, 7 June 80, Northern Ireland

The outlawed Irish Republican Army shot dead a part-time soldier in front of his 11-year-old son in a village store Saturday.

IPP OUTPUT: "Terrorist attacks in Northern Ireland are carried out by members of the Irish Republican Army."

UPI, 12 February 80, Italy

Red Brigades guerrillas firing silenced pistols killed a prominent judge with strong ties to the Vatican while he was at a conference on terrorism today.

UPI, 12 March 80, Italy

Gunmen using silencer-equipped pistols killed a right-wing political activist today in Italy's 18th reported political assassination of the year.

IPP OUTPUT: "Pistols with silencers are a common weapon for shootings in Italy."

New York Times, 9 December 79

The police today freed a kidnapped industrialist and arrested two men who had held him prisoner in an apartment near Como.

Francesco Massoni, 75 years old, had been abducted four days ago from his home in Pavia, 40 miles to the south.

Washington Post, 23 December 79

The owner of a metal works in Northern Italy was abducted late Friday night in the country's 68th reported kidnapping of the year.

IPP OUTPUT: "The victims of kidnappings in Italy are usually businessmen."

UPI, 19 January 80, Italy

Urban guerrillas bombed a police station on the edge of Rome early Saturday causing heavy damage and injuring 17 officers sleeping inside.

UPI, 17 February 80, Italy

Urban guerrillas Sunday night bombed an electrical power station in the southern part of the city where the ruling Christian Democratic Party is holding its 14th national convention.

IPP OUTPUT: "Bombings in Italy are usually carried out by urban terrorists."

UPI, 27 February 80, Colombia

Heavily armed leftist guerrillas shot their way into a diplomatic reception Wednesday, seized the US ambassador and 44 other hostages, and threatened to "start the painful task of executing them" unless the army pulled back from the diplomatic compound.

UPI, 7 May 80, El Salvador

Leftist guerrillas seized two San Salvador radio stations Wednesday afternoon and broadcast 20-minute tape recorded messages urging the public to join the guerrillas or donate money to their cause.

IPP OUTPUT: "Building takeovers in Latin America are usually done by left-wing groups".

IPP processed stories from newspapers and the UPI newswire using its own generalizations to better understand future stories. When a story was recognized as being similar to previous stories, IPP attempted to make generalizations that accounted for similarities among the events. IPP could also confirm its generalizations as more stories were read.

IPP used two knowledge structures to organize its memory: the S-MOP (Simple Memory Organization Packet) and the action unit. Based on the concept of MOPs (see Memory-organization packets) (16), an S-MOP describes abstract situations that can be used to explain widely varying physical activities. For example, the concept of extortion may be used to describe a variety of events including kidnappings, hijackings, and blackmail. An action unit is a structure that organizes concrete events much like a script. The major difference between scripts as they were used in SAM and action units in IPP is one of intended utility. SAM's scripts were used to generate inferences by looking inside the scriptal knowledge structure at its event sequences. But action units are useful in IPP as "primitives"—indivisible clusters of events that can be manipulated by higher level processes. By bundling complex events like kidnappings and hijackings into uniform structures, IPP was able to operate without dropping down into the internal structure of these concepts. If needed, each action unit could be decomposed into conceptual-dependency representations (5), goals (6), political acts (17), and other action units. But the more important information associated with an action unit was its relationships to S-MOPs.

Each of the systems described so far was designed to investigate specific issues in text comprehension. SAM, PAM, and

POLITICS explored fundamental problems associated with knowledge structures and predictive processing. FRUMP applied many of these knowledge-based processing techniques in an effort to show how a system could effectively skim text to extract only the most important information. IPP went one step further in showing how a rudimentary type of learning could take place by incorporating processes of generalization. Yet with each new experiment, certain patterns became apparent.

On an intuitive level, it became obvious that sophisticated systems require a lot of knowledge—not just brute-force quantities of knowledge but many different kinds of knowledge as well. The best script-applier mechanism in the world equipped with all of the scripts anyone could imagine would never be able to handle the simplest stories understood by PAM because PAM operated with a qualitatively different kind of knowledge.

A less intuitive but nevertheless significant step was taken by FRUMP and IPP in terms of overall system architecture. Although the earlier text-processing systems all operated with a separate sentence analyzer acting as the front-end, FRUMP and IPP integrated sentence analysis with processes that accessed and manipulated memory. The line between understanding a sentence and drawing inferences was gradually becoming very fuzzy.

### Memory Organization

With all of this work serving as a backdrop, Dyer set out to build an ambitious story-understanding system that would draw from multiple knowledge structures of various types as well as explore the question of memory integration as a problem in sentence analysis. The result was BORIS (18). In addition to using the standard knowledge structures introduced by other systems, BORIS went on to incorporate thematic abstraction units, interpersonal themes, and knowledge about affective reactions. With this impressive arsenal of knowledge at its disposal, BORIS could tackle texts resembling short stories, although it was never intended to handle a large number of stories robustly.

BORIS used 17 different types of knowledge structures. It was crucial to formulate some systematic method for knowledge structure interactions given the large number of potential interactions possible. To solve this problem, BORIS used a "knowledge-dependency graph," which cuts the number of potential interactions down to 28. These pairwise interactions are represented in Fig. 4 as links in the knowledge-dependency graph.

BORIS was also designed to integrate processes of sentence analysis with processes of inference and memory management. In BORIS a sentence cannot be understood without integrating its meaning into the larger memory representation being built for the story. This resulted in analysis strategies that were highly sensitive to the surrounding context of previous text.

One of the most important contributions to be found in BORIS is a new knowledge structure called a TAU (Thematic Abstraction Unit). Each TAU can be associated with a familiar adage, such as "out of the frying pan and into the fire" or "counting your chickens before they hatch." Internally, a TAU is a knowledge structure based on planning failures. Since people often use adages to describe the ultimate meaning of a story, it is important to recognize standard TAUs when they

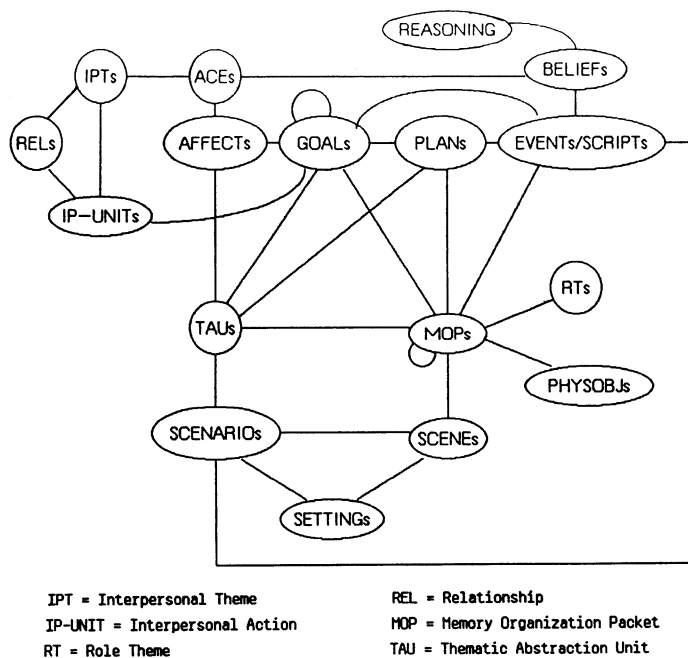


Figure 4. Knowledge-dependency graph.

arise. Aside from providing one with a model for impressive summarization behavior, TAU's are also used predictively as another source of inference at the time of understanding.

Although BORIS was implemented to demonstrate its comprehension by answering questions about its stories, ideas concerning narrative summarization were inspired by the efforts behind BORIS. TAU's were useful for a very high level of abstraction, but people can also summarize stories more concretely, without reference to standard adages at all. Representational work on affect for the BORIS system led to a proposal for plot units (19) as memory structures that are useful for text summarization at a more concrete level of actual events and characters.

Plot units are configurations of mental states describing stereotypical interactions between characters or simple mental state changes within a single character. By extracting information from goal states and planned behavior, plot units can be recognized and instantiated for all major characters in a narrative. A plot-unit graph is then constructed that shows how the various plot units interact with one another in terms of their interconnectedness.

A system named PUGG was designed to generate plot-unit graphs for narrative texts in order to identify the most important events in a narrative (20). Each plot-unit graph was constructed of nodes representing instantiated plot units with arcs joining two nodes whenever their underlying plot units shared a common mental state. Structural properties of the resulting plot-unit graph then determine which nodes will point to information needed for a good summary. In small graphs a unique node of maximal degree is expected to represent the central event of the story. In larger graphs a path between cut points of the graph may be the key to a reasonable summary.

PUGG was used to analyze stories ranging in length from one paragraph to several pages. In its most ambitious test

PUGG generated a plot-unit graph for the New Testament, based on Arnold Toynbee's synopsis of the story of Jesus (21). The resulting graph contained over 100 nodes, and a cut-point algorithm was used to identify key plot units for a narrative summary. The concepts PUGG identified as being central to the story provided a remarkably concise and satisfactory summary:

Jesus makes an appeal to the masses for support.

The government wants to maintain authority over the masses.

Jesus causes a scandal.

Jesus takes the law into his own hands to avenge God.

The authorities arrest Jesus.

Jesus is crucified.

Jesus' death is a triumph.

Jesus is worshipped.

It is nevertheless important to note that PUGG could not construct a plot-unit graph unless a system like BORIS had analyzed the source text first and constructed intermediate layers of memory representation based on scripts, plans, goals, etc. In the case of the New Testament example, these intermediate memory levels were hand-coded since no story-understanding system has access to the amount of knowledge needed to handle texts as involved as the New Testament (or even Arnold Toynbee's synopsis of it). In theory, it would be possible to encode the knowledge structures needed, but it would take more programmers and more time than is generally considered appropriate for a research effort.

Although there are still plenty of theoretical problems to be solved in the area of knowledge-based text understanding, one can appreciate the more pragmatic problems involved with making large amounts of knowledge available to a computer. Many researchers argue that it will not be possible to hand-code all of the knowledge structures needed for a general text-understanding system; it is crucial to develop learning capabilities of the sort envisioned by Lebowitz (15). Other researchers suggest that the current technology is appropriate for circumscribed domains where a relatively small amount of knowledge is adequate; it may never be possible to build a general story understander that operates without domain limitations. Whatever the outcome, AI research in story understanding has shed light on the complexities of human cognition and given a greater appreciation for the mundane information-processing skills of average people.

## BIBLIOGRAPHY

1. C. Rieger, Conceptual Memory and Inference, in R. Schank (ed.), *Conceptual Information Processing*, American Elsevier, New York, pp. 157-288, 1975.
2. A. Graesser, *Prose Comprehension Beyond the Word*, Springer-Verlag, New York, 1981.
3. R. Cullingford, Script Application: Computer Understanding of Newspaper Stories, Research Report No. 116, Department of Computer Science, Yale University, New Haven, CT, 1978.
4. R. Cullingford, SAM, in R. Schank and C. Riesbeck (eds.), *Inside Computer Understanding: Five Programs Plus Miniatures*, Erlbaum, Hillsdale, NJ, pp. 75-119, 1981.

5. R. Schank, *Conceptual Information Processing*, North-Holland, Amsterdam, 1975.
6. R. Schank and R. Abelson, *Scripts, Plans, Goals and Understanding*, Erlbaum, Hillsdale, NJ, 1977.
7. W. Lehnert, *The Process of Question Answering*, Erlbaum, Hillsdale, NJ, 1978.
8. W. Lehnert, The Role of Scripts in Understanding, in D. Metzger (ed.), *Frame Conceptions and Text Understanding*, de Gruyter, New York, pp. 79–95, 1980.
9. V. Abbott and J. Black, The Representation of Scripts in Memory, Cognitive Science Technical Report No. 5, Yale University, August 1980.
10. G. H. Bower, J. B. Black, and T. Turner, "Scripts in memory for text," *Cog. Psychol.* 11, 177 (1979).
11. G. DeJong, "Prediction and substantiation: A new approach to natural language processing," *Cog. Sci.* 3(3), 251–273 (1979).
12. J. Carbonell, "POLITICS: Automated ideological reasoning," *Cog. Sci.* 2(1), 27–51 (1978).
13. R. Wilensky, PAM, in R. Schank and C. Riesbeck (eds.), *Inside Computer Understanding: Five Programs Plus Miniatures*, Erlbaum, Hillsdale, NJ, pp. 136–179, 1981.
14. R. Wilensky, *Planning and Understanding: A Computational Approach to Human Reasoning*, Addison-Wesley, Reading, MA, 1983.
15. M. Lebowitz, "Generalization from natural language text," *Cog. Sci.* 7(1), 1–40 (1983).
16. R. Schank, *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*, Cambridge University Press, Cambridge, UK, 1982.
17. R. Schank and J. Carbonell, Re: The Gettysburg Address: Representing Social and Political Acts, Research Report No. 127, Department of Computer Science, Yale University, New Haven, CT, 1978.
18. M. Dyer, *In-Depth Understanding: A Computer Model of Integrated Processing for Narrative Comprehension*, MIT Press, Cambridge, MA, 1983.
19. W. Lehnert, Plot Units: A Narrative Summarization Strategy, in Lehnert and Ringle (eds.), *Strategies for Natural Language Processing*, Erlbaum, Hillsdale, NJ, pp. 375–412, 1982.
20. W. Lehnert and C. Loiselle, Plot Unit Recognition For Narratives, Technical Report No. 83-39, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, 1983.
21. W. Lehnert, Narrative Complexity Based on Summarization Algorithms, in B. Bara and G. Guida (eds.), *Computational Models of*

*Natural Language Processing*, Elsevier Science, Amsterdam, 1984.

W. G. LEHNERT  
University of Massachusetts

## STRIPS

A problem-solving (qv) system designed by Fikes and Nilsson at Stanford Research Institute, STRIPS strives to find a solution sequence of operators to transform the initial model into the goal state [see R. Fikes and N. Nilsson, "STRIPS: a new approach to the application of theorem proving to problem solving," *Artif. Intell.* 2, 189–208 (1971)].

A. HANYONG YUHAN  
SUNY at Buffalo

## STUDENT

Bobrow's STUDENT solves algebra story problems given in a subset of English (see Problem solving). It transforms natural language into a set of simpler sentences that are equivalent in meaning, so-called kernel sentences. These are then transformed into a set of simultaneous equations. STUDENT was tested on problems from schoolbooks (after adapting them to STUDENT's limited subset of English), and it solved them as fast or faster than humans. A complete description of STUDENT can be found in Bobrow's dissertation (D. G. Bobrow, *Natural Language Input for a Computer Problem-Solving System*, Report TR-1, Project MAC, MIT, Cambridge, MA, 1964) which was later reprinted [D. G. Bobrow, *Natural Language Input for a Computer Problem-Solving System*, in M. Minsky, (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, pp. 146–226, 1968].

J. GELLER  
SUNY at Buffalo

**SYMBOL PROCESSING.** See LISP, Programming environments; PROLOG.

**SYNTACTIC PARSING.** See Parsing.

**TACTILE SENSING.** See Sensors; Robotics.

**TARGET TRACKING.** See Military, applications in.

**TASK-ORIENTED LANGUAGE.** See Robotics.

## TELEOPERATORS

Current practice in robotics (qv) divides into two main areas: industrial robotics and robotic teleoperation. Industrial robots are used as an integral part of manufacturing processes and within the frame of production-engineering techniques to perform repetitive work in a structured factory environment (see Robots, mobile). The characteristic control of industrial robots is a programmable sequence controller, typically a mini- or microcomputer that functions autonomously with only occasional human intervention, either to reprogram or retool for a new task or to correct for an interruption in the work flow. Teleoperator robots, on the other hand, serve to extend, through mechanical, sensing, and computational techniques, the human manipulative, perceptive, and cognitive abilities into an environment that is either hostile to or remote from the human operator. Teleoperator robots, or in today's nomenclature, "telerobots," typically perform nonrepetitive (singular) servicing, maintenance, or repair work under a variety of environmental conditions ranging from structured to unstructured conditions. Telerobot control is characterized by a direct involvement of the human operator in the control since, by definition of task requirements, teleoperator systems extend human manipulative, perceptual, and cognitive skill which is far beyond what is obtainable with today's industrial robots.

Applications of teleoperators are numerous, in particular to the nuclear and munitions industries and in explorations that may pose danger to human life. Most teleoperators in use today are static master-slave manipulators with sole mechanical-control connection between master and slave arms, and the master arm is a replica of the slave arm. When large distances prevent a sole mechanical-control connection, the master and slave units are connected through electromechanical servo-mechanisms. Only few working systems use a digital-control communication link between master and slave arms. The master-slave-manipulator systems embody a very useful property in teleoperation: bilateral or force-reflecting control, which permits the human operator to feel the forces and moments acting on the remote slave arm and hand while he/she manually controls the motion of the slave arm. In this control mode the operator is kinematically and dynamically "coupled" to the remote robot arm and can command with "feel" and control with a "sense of touch." State-of-the-art teleoperator technology, including terminology, categories of teleoperators, teleoperator design context, and teleoperator performance predictions are summarized in Ref. 1. Most of the existing manipulators employed in teleoperation, together with their design features are listed in Ref. 2, and human considerations in teleoperation are reviewed in Ref. 3.

Continuous human operator control in teleoperation has both advantages and disadvantages. The main advantage is that overall task control can rely on human perception, judgment, decision, dexterity, and training. The main disadvantage is that the human operator must cope with a sense of remoteness, be prepared for integration of many information and control variables, and coordinate the control of one or two mechanical arms each having many (typically six) degrees of freedom—and doing all these with limited human resources. Furthermore, in many cases, such as space and deep-sea applications, communication time delay interferes with continuous human operator control (4).

Modern development trends in teleoperator control technology are aimed at amplifying the advantages and alleviating the disadvantages of the human element in teleoperator control through the development and use of advanced sensing and graphics displays, intelligent computer controls, and new computer-based man-machine-interface devices and techniques in the information and control channels. The use of model-driven and sensor-data-driven automation in teleoperation offers significant new possibilities to enhance overall task performance by providing efficient means for task-level controls and displays as outlined and exemplified in Refs. 5 and 6. Manipulators on a mobile base represent new demands for sensor-based automation in teleoperation as discussed and exemplified in Ref. 7.

Automation (qv) in teleoperation is distinguished from other forms of automated systems by the explicit and active inclusion of the human operator in system control and information management. Such active participation by the human, interacting with automated system elements in teleoperation, is characterized by several levels of control and communication and can be conceptualized in the category of "supervisory control," as discussed in Ref. 8. The man-machine interaction levels in teleoperation can be considered in a hierarchic arrangement as outlined in Ref. 9: planning or high-level algorithmic functions; motor or actuator control functions; and environmental interaction-sensing functions. These functions take place in a task context in which the level of system automation is determined by the mechanical and sensing capabilities of the telerobot system; real-time constraints on computational capabilities to deal with control, communication, and sensing; the amount, format, content, and mode of operator interaction with the telerobot system; environmental constraints, such as task complexity, and overall system constraints, such as operator's skill or maturity of machine intelligence techniques.

Some advances have been made in teleoperator technology through the introduction of various sensors, computers, automation, and new man-machine-interface devices and techniques for remote manipulator control. The development of dexterous mechanisms, smart sensors, flexible computer controls, intelligent man-machine interfaces, and innovative system designs for advanced teleoperation is, however, far from complete and poses many interdisciplinary challenges (10) (see Human-Computer interaction). It should also be recognized that the normal manual dexterity of humans is more a "body" skill than an intellectual one. The man-machine-inter-



face philosophy embodied in the force-reflecting master-slave manipulator-control technology has been founded mainly on this fact. Advanced teleoperation employing sensor-referenced and computer-controlled manipulators shifts the operator-telerobot interface from the body (analog) level to a more intellectual languagelike (symbolic) level. Research efforts for developing new man-machine-interface technology for advanced teleoperation will have to render the languagelike symbolic interface between human operator and telerobot as efficient as the conventional analog interface. This remark also applies to operator-interface development for procedure-execution aids and for expert systems in teleoperator action planning and error recovery.

## BIBLIOGRAPHY

1. W. J. Book, Teleoperator Arm Design, in S. Y. Nof (ed.), *Handbook of Industrial Robotics*, Wiley, Chapter 9, pp. 138-157, 1985.
2. G. W. Kohler, *Manipulator Type Book*, Verlag Karl Thiemeig, Munich, FRG, 1981.
3. E. G. Johnson and W. R. Corliss, *Human Factors Applications in Teleoperator Design and Operation*, Wiley, New York 1971.
4. E. Heer (ed.), *Remotely Manned Systems: Explorations and Operations in Space*, California Institute of Technology, Pasadena, CA, 1973.
5. A. K. Bejczy, Control of Remote Manipulators, in S. Y. Nof (ed.), *Handbook of Industrial Robotics*, Wiley, New York, Chapter 17, pp. 320-333, 1985.
6. S. Lee, G. Bekey, and A. K. Bejczy, Computer Control of Space-Borne Teleoperators with Sensory Feedback, *Proceedings of the IEEE International Conference on Robotics and Automation*, St. Louis, MO, March 25-28, 1985, pp. 205-214.
7. C. M. Witkowski, A. H. Bond, and M. Burton, The Design of Sensors for a Mobile Teleoperator Robot, *Digital Systems for Industrial Automation*, 2(1), 85-111 (1983).
8. T. B. Sheridan, Modeling Supervisory Control of Robots, in A. Morecki and K. Kedzior (eds.), *Theory and Practice of Robots and Manipulators*, Elsevier Scientific, New York, pp. 394-406, 1977.
9. A. K. Bejczy and K. Corker, Automation in Teleoperation from a Man-Machine Interface Viewpoint, *Proceedings of AIAA/NASA Space Systems Technology Conference*, Costa Mesa, CA, June 5-7, 1984, pp. 87-95.
10. A. K. Bejczy, "Sensors, controls, and man-machine interface for advanced teleoperation," *Science* 208, 1327-1335 (1980).

A. K. BEJCZY  
Jet Propulsion Laboratory

This work has been carried out under contract with the National Aeronautics and Space Administration.

**TEMPLATE MATCHING.** See Matching, template.

**TEXT PROCESSING.** See Natural-language generation; Natural-language understanding; Parsing.

## TEXTURE ANALYSIS

**Notion of Texture.** The world possesses hierarchically complex structure. If one looks closely at what appears to be a simple object, one discovers additional structural complexity

that escaped detection earlier. Similarly, what appears to be a configuration of distinct objects may compress into a single perceived component of the scene when a larger visual field is viewed. The perception of distinct components in a given visual field is usually determined by the limits of the viewer's resolving power (see Image understanding).

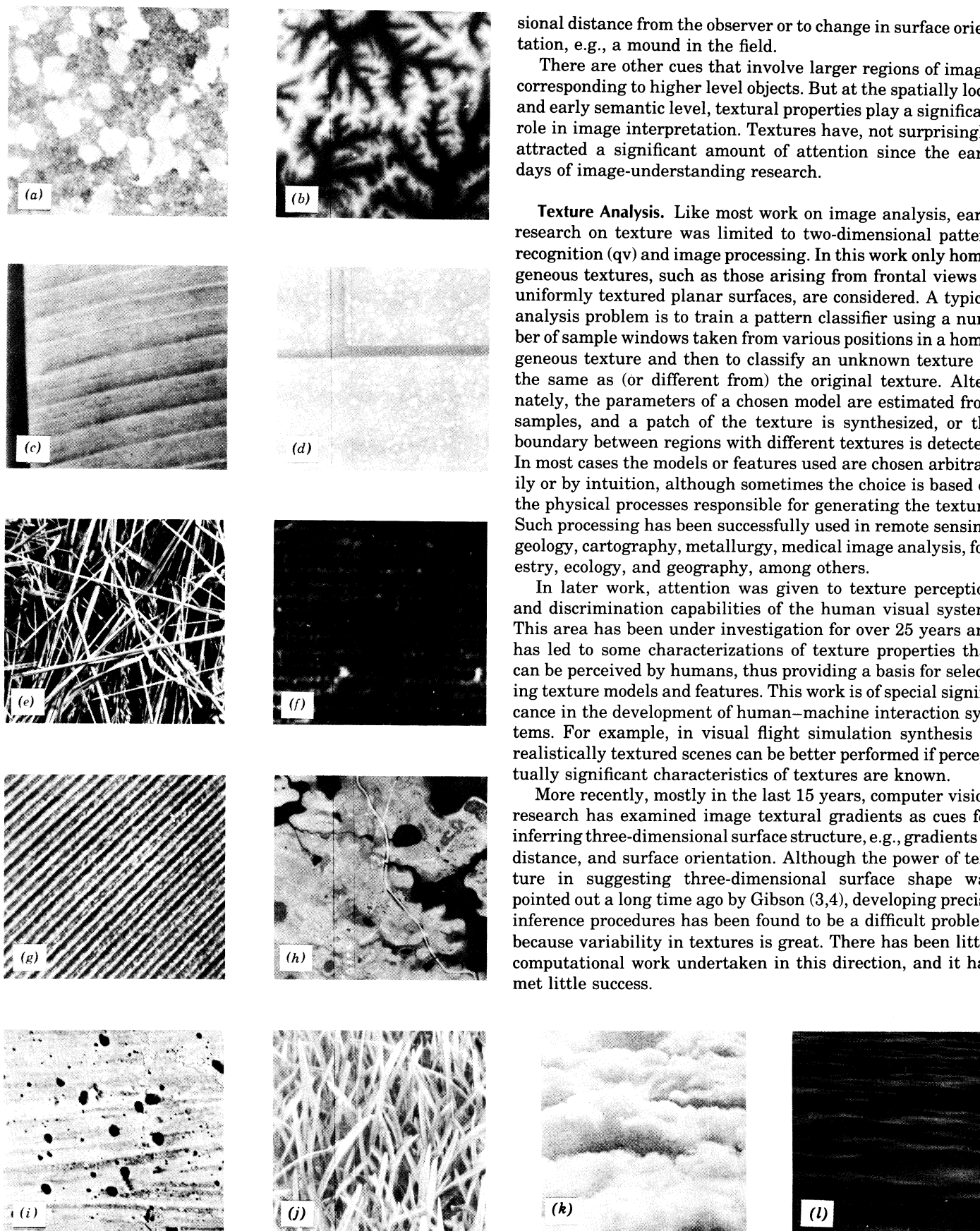
When interpreting an image, thus, one has access to image components that represent perceivable scene components. Although these components may correspond to high-level objects, they consist of structural primitives characterized only by low-level, or syntactic, homogeneity. For example, a chair may well be perceived as a component of the scene, but the region in the image occupied by the chair consists of homogeneous subregions laid out in a certain arrangement.

Before any higher level inferences/interpretations can be made, the large amount of image data must be reduced into a description in terms of structural primitives that mark the starting point for synthesis of scene perception. What are these primitive regions? They arise physically because of limits on spatial resolution. For example, a region may be perceived to be homogeneous because the eye cannot distinguish among the colors of small individual subregions and sees the whole region having a single color. Alternately, a region may be perceived to possess uniform intensity.

A third type of homogeneity of a region comes from a high structural-packing density. Consider, e.g., an image showing a patch of grass. There is so much structural information present in the image that one is not aware of each individual grass blade. In spite of this, the uniformity of the region is well perceived in the sense that if there were a bush with thousands of leaves in the middle of the grass patch, it would be spontaneously discriminable from the grass even if the bush had exactly the same color as the grass.

This kind of spatial homogeneity is attributed to uniformity of texture (see Fig. 1 for examples of textures). Clearly, texture is a macrostructure of image regions. Beyond this intuitive notion, however, it is hard to give a precise definition of texture. Picket (1) views texture as consisting of a large number of elements, each in some degree visible and, on the whole, densely and evenly (possibly randomly) distributed over the field of view such that there is a distinct, characteristic, spatial repetitiveness in the pattern. Texture has been characterized by attributes of surface quality that it evokes when a region carrying a given texture is viewed. Thus, properties such as fine, coarse, smooth, granulated, rippled, mottled, irregular, random, and lineated have been used to characterize textures (2).

**Role of Texture In Visual Perception.** The myriad of combinations of pixel colors and intensities possible in an image would pose serious computational problems for visual perception if not for the redundancy present in pixel information. Visual interpretation depends on certain macrocharacteristics of pixel values, or visual cues (3,4). Some cues concern the nature of continuous variation of color or intensity, e.g., increasing bluishness or blurriness is interpreted as increasing distance in the scene, and variation in shading is attributed to variation in surface orientation. Other monocular image cues concern spatial variation in textural properties. For example, an image region showing a cornfield may be easily discriminated from an adjacent region showing a wheat field. The spatial variation in textural properties of an image showing a grass field in perspective is attributed to change in three-dimen-



**Figure 1.** Some real textures. (a) Volcanic rock; (b) terrain elevations; (c) leaf surface; (d) brick wall; (e) straw; (f) orchard; (g) plowed field; (h) mud tidal flats; (i) cement; (j) grass (in perspective); (k) cloud layer (in perspective); (l) water waves (in perspective) (from Ref. 139).

sional distance from the observer or to change in surface orientation, e.g., a mound in the field.

There are other cues that involve larger regions of images corresponding to higher level objects. But at the spatially local and early semantic level, textural properties play a significant role in image interpretation. Textures have, not surprisingly, attracted a significant amount of attention since the early days of image-understanding research.

**Texture Analysis.** Like most work on image analysis, early research on texture was limited to two-dimensional pattern recognition (qv) and image processing. In this work only homogeneous textures, such as those arising from frontal views of uniformly textured planar surfaces, are considered. A typical analysis problem is to train a pattern classifier using a number of sample windows taken from various positions in a homogeneous texture and then to classify an unknown texture as the same as (or different from) the original texture. Alternately, the parameters of a chosen model are estimated from samples, and a patch of the texture is synthesized, or the boundary between regions with different textures is detected. In most cases the models or features used are chosen arbitrarily or by intuition, although sometimes the choice is based on the physical processes responsible for generating the texture. Such processing has been successfully used in remote sensing, geology, cartography, metallurgy, medical image analysis, forestry, ecology, and geography, among others.

In later work, attention was given to texture perception and discrimination capabilities of the human visual system. This area has been under investigation for over 25 years and has led to some characterizations of texture properties that can be perceived by humans, thus providing a basis for selecting texture models and features. This work is of special significance in the development of human-machine interaction systems. For example, in visual flight simulation synthesis of realistically textured scenes can be better performed if perceptually significant characteristics of textures are known.

More recently, mostly in the last 15 years, computer vision research has examined image textural gradients as cues for inferring three-dimensional surface structure, e.g., gradients of distance, and surface orientation. Although the power of texture in suggesting three-dimensional surface shape was pointed out a long time ago by Gibson (3,4), developing precise inference procedures has been found to be a difficult problem because variability in textures is great. There has been little computational work undertaken in this direction, and it has met little success.

Most of the research on texture analysis and its applications has been in the first of the three areas mentioned above. This is so partly because of the long time over which the work in the first area has been carried out but mainly because it is the simplest among the three areas considered. Consequently, a major portion of this review is devoted to the work on modeling of textures as homogeneous, two-dimensional patterns. Such models have traditionally been categorized as either statistical or structural. Statistical models characterize an image in terms of its statistical properties, such as autocorrelation or cooccurrence. Structural models, on the other hand, describe an image by its structural primitives and their placement rules. This classification is not very useful, though, for if a structural model is not also statistical, the images it describes are too regular to be of interest. If a statistical model cannot reveal the image's basic structure, it is too weak to be of much help. A somewhat better division of image models into pixel-based and region-based models is used by Ahuja and Schachter (5).

This entry discusses the pixel-based and region-based models, reviews some work on models of textures proposed for human visual perception, describes the work done on extracting three-dimensional shape from image texture, and presents concluding remarks.

### Pixel-Based Models

These models view individual pixels as the primitives of an image. Specification of the characteristics of the spatial distribution of pixel properties (6,7) constitutes the image description.

Pixel-based models can be subdivided into two classes: one-dimensional time-series models and random-field models.

**One-Dimensional Time-Series Models.** Time-series analysis (8) has been extensively used to model the statistical relationship between the gray level of a given pixel and of those preceding it in the raster scan (9–11). The gray-level fluctuations along the raster are treated as a stochastic process that evolves over time. The future course of the process is presumed to be predictable from information about its past.

Following is some commonly used notation in these models (9). Let

$$\cdots X_{t-1} X_t X_{t+1} \cdots$$

be a discrete time series where  $X_i$  is the random variable  $X$  at time  $i$ . The series is denoted by  $[X]$ .

Let  $\mu$  be the mean of  $[X]$ , called the level of the process.

Let  $[\tilde{X}]$  denote the series of deviations about  $\mu$ , i.e.,  $\tilde{X}_i = X_i - \mu$ .

Let  $[e]$  be a series of outputs from a white noise source with mean zero and variance  $\sigma^2$ .

Let  $B$  be the "backward" shift operator for the deviation series such that  $B\tilde{X}_t = \tilde{X}_{t-1}$ ; hence  $B^m\tilde{X}_t = \tilde{X}_{t-m}$ .

Let  $\nabla$  be the backward difference operator for the deviation series such that

$$\nabla\tilde{X}_t = \tilde{X}_t - \tilde{X}_{t-1} = (1 - B)\tilde{X}_t$$

$$\text{Hence } \nabla^m\tilde{X}_t = (1 - B)^m\tilde{X}_t.$$

The dependence of the current value  $\tilde{X}_t$  on the past values of  $\tilde{X}$  and  $e$  can be expressed in different ways, giving rise to several different models (9).

(a) *Autoregressive Model* (AR). In this model the current  $\tilde{X}$ -value depends on the previous  $P$   $\tilde{X}$ -values and on the current noise term

$$\tilde{X}_t = \phi_1\tilde{X}_{t-1} + \phi_2\tilde{X}_{t-2} + \cdots + \phi_p\tilde{X}_{t-p} + e_t \quad (1)$$

Let

$$\phi_p(B) = 1 - \phi_1B - \phi_2B^2 - \cdots - \phi_pB^p$$

Then Eq. 1 becomes

$$[\phi_p(B)](\tilde{X}_t) = e_t$$

The series  $[\tilde{X}]$ , as defined above, is known as the autoregressive process of order  $p$ , and  $\phi_p(B)$  as the autoregressive operator of that order  $p$ . The name "autoregressive" comes from the model's similarity to regression analysis and the fact that the variable  $\tilde{X}$  is being regressed on previous values of itself.

(b) *Moving-Average Model* (MA). In (a) above,  $\tilde{X}_{t-1}$  can be eliminated from the expression for  $\tilde{X}_t$  by substituting

$$\tilde{X}_{t-1} = \phi_1\tilde{X}_{t-2} + \phi_2\tilde{X}_{t-3} + \cdots + \phi_p\tilde{X}_{t-p-1} + e_{t-1}$$

This process can be repeated to eventually yield an expression for  $\tilde{X}_t$  as an infinite series in the  $e$ 's.

A moving-average model allows a finite number  $q$  of previous  $e$ -values in the expression for  $\tilde{X}_t$ . This explicitly treats the series as being observations on linearly filtered Gaussian noise. Letting

$$\theta_q(B) = 1 - \theta_1B - \theta_2B^2 - \cdots - \theta_qB^q$$

one has

$$\tilde{X}_t = [\theta_q(B)](e_t)$$

as the moving-average process of order  $q$ .

(c) *Mixed Model: Autoregressive/Moving Average* (ARMA). To achieve greater flexibility in the fitting of actual times series, this model includes both the autoregressive and the moving-average terms. Thus,

$$\tilde{X}_t = \phi_1\tilde{X}_{t-1} + \phi_2\tilde{X}_{t-2} + \cdots + \phi_p\tilde{X}_{t-p} + e_t - \theta_1e_{t-1} - \theta_2e_{t-2} - \cdots - \theta_qe_{t-q} \quad (2)$$

that is,  $[\phi_p(B)](\tilde{X}_t) = [\theta_q(B)](e_t)$ .

In all three models just mentioned, the process generating the series is assumed to be in equilibrium about a constant mean level. Models characterized by this equilibrium are called stationary models.

There is another class of models, nonstationary models, in which the level  $\mu$  does not remain constant. The series involved may, nevertheless, exhibit homogeneous behavior after the differences due to level drift have been accounted for. It can be shown (8) that such behavior may be represented by a generalized autoregressive operator.

A time series may exhibit a repetitive pattern. For example, in a raster-scanned image the segments corresponding to rows will have similar characteristics. A model can be formulated that incorporates such "seasonal effects" (9,12).

All of the time-series models discussed above are unilateral in that a pixel depends only on the pixels preceding it in the raster scan. Partial two-dimensional dependence among pixels can be incorporated by letting a pixel depend on a causal neighborhood (10–13), e.g., on that part of an  $n \times n$  neighbor-

hood centered at the pixel that precedes it in the raster scan (13). This does not affect the applicability of the one-dimensional time-series analysis. Whittle (13) points out that a one-dimensional approach has serious deficiencies. For example, even a finite bilateral autoregression may not always have a unilateral representation that is also a finite autoregression. Or the transformation to convert a bilateral dependence into a unilateral one may be prohibitively complex. Introduction of bilateral dependence gives rise to more complex parameter-estimation problems (14,15). A frequency-domain treatment makes parameter estimation in bilateral representation much easier (16)].

**Random-Field Models.** The theory of stochastic processes may be extended to define models of spatial variation in two dimensions. These models can be divided into two subclasses: global models and local models.

**Global Models.** Global models treat an entire image as the realization of a random field. The most common approach is to view an image as an ideal signal that has been corrupted by blurring and additive noise. The blur may represent a variety of spatial degradations (17–19). For example, in aerial reconnaissance, astronomy, and remote sensing, the pictures obtained are degraded by atmospheric turbulence, aberrations of the optical system, and relative motion between object and camera. Electron micrographs are affected by the spherical aberration of the electron lens. Additive noise represents degradations that only affect the gray levels of individual points. A common example is thermal noise occurring in photodetectors. Film grain noise, which is multiplicative, can be converted to additive noise by subjecting the image to a logarithmic transformation.

Frieden (20) describes image restoration using global models. The problem involves solving the imaging equation

$$\mathbf{D} = \mathbf{S}\mathbf{X} + \mathbf{n}$$

for the ideal image row  $\mathbf{X}$ , given the image data  $\mathbf{D}$  and an estimate of the point spread matrix  $\mathbf{S}$ , despite the presence of an additive noise component  $\mathbf{n}$ . Here  $\mathbf{X}$ ,  $\mathbf{D}$ , and  $\mathbf{n}$  are column vectors of  $n$  elements each, and  $\mathbf{S}$  is an  $n \times n$  matrix. Although inversion of the image formation equation is an unstable or ill-conditioned problem, a reasonable solution can be obtained with a priori knowledge of the nature of the true image. This information takes the form of constraints in the restoration procedure. Frieden treats the image as a spatial distribution of photons in cells centered at the grid nodes and then defines the image model as the process most likely to have generated the given distribution.

For restoration, images have often been modeled as two-dimensional, wide-sense, stationary random fields having a given mean and autocorrelation function. The following general expression has been suggested for the autocorrelation function:

$$R(\tau_1, \tau_2) = \sigma^2 \rho^{[-\alpha_1|\tau_1| - \alpha_2|\tau_2|]}$$

which is stationary and separable. Specifically, the exponential autocorrelation function ( $\rho = e$ ) has been found to be useful for a variety of pictorial data (21–25).

Another autocorrelation function often cited as being more realistic is

$$R(\tau_1, \tau_2) = \rho \sqrt{\tau_1^2 + \tau_2^2}$$

which is isotropic, but not separable, as is the case with many natural images.

Hunt (26,27) points out that stationary Gaussian models are based on an oversimplification. Consider the vector  $\mathbf{x} = (x_1, \dots, x_n)$  formed from sample values along a raster scan, with  $R$  being the covariance matrix of the gray levels in  $\mathbf{x}$ . According to the Gaussian assumption, the probability density function is given by

$$f(\mathbf{x}) = k \exp\{-\frac{1}{2}(\mathbf{x} - \mu)R^{-1}(\mathbf{x} - \mu)\}$$

where  $\mu = E[\mathbf{x}]$ , and  $k$  is a normalizing constant. The stationarity assumption makes  $\mu$  a vector of identical components, meaning that each point in the image has identical ensemble statistics. Hunt (27) proposes a nonstationary Gaussian model that differs from the stationary model only in that the mean vector  $\mu$  has unequal components. He demonstrates the appropriateness of this model by subtracting the local ensemble average from each image point and showing that the resulting image fits a stationary Gaussian model.

In a later paper Hunt (28) discusses three different kinds of nonstationarities:

*Case 1:* Nonstationary mean, nonstationary autocorrelation.

*Case 2:* Nonstationary mean, stationary autocorrelation.

*Case 3:* Stationary mean, nonstationary autocorrelation.

Since a breakdown in stationarity implies a loss of the ergodicity assumption [loosely stated, a process is called “ergodic” if its statistics can be determined from any of the sample functions in the ensemble (29)], it is necessary to specify image statistics in terms of spatial averages rather than ensemble averages. The nonstationarity in the image mean is described by the array of individual means taken over a specified neighborhood about each image point. For the autocorrelation function, the breakdown of stationarity is related to the way the correlation function changes over the image. Hunt uses three attributes of the correlation function to describe its spatial dependence: its energy, its width, and its shape. An image model consists of a specification of the parameters of a rotationally symmetric, negative exponential autocorrelation function, a set of means, and a spatial warp function to produce the autocorrelation nonstationarity.

Trussel and Kruger (30) claim that the Laplacian density function is better suited for modeling high-pass filtered imagery than the Gaussian function. Nevertheless, they contend that the basic assumptions that allow the Gaussian model to be used for image-restoration purposes are still valid under a Laplacian model.

Nahi and Jahanshahi (31) suggest modeling the image by background and foreground statistical processes. The foreground consists of regions corresponding to the objects in the image. Each type of region (foreground or background) is defined by an associated statistical process. In estimating the boundaries of horizontally convex objects in noisy binary images, Nahi and Jahanshahi assume that the two processes are statistically independent stationary random processes with known (or estimated) first two moments. The borders of the regions covered by the different statistical processes are modeled only locally. The end points of the intercepts of the given object on successive rows are assumed to form a first-order

Markov process. This model thus also involves local interactions.

Using the notation

- $x_i$  = the gray level at image point  $i$ ,
- $\gamma_i$  = a binary function carrying the boundary information at point  $i$ ,
- $b_i$  = a sample gray level from the background process at point  $i$ ,
- $o_i$  = a sample gray level from the foreground process at point  $i$ ,
- $e_i$  = a sample gray level from the noise process at point  $i$ ,

the model can be written as

$$x_i = \gamma_i o_i + [1 - \gamma_i] b_i + e_i$$

where  $\gamma$  incorporates the Markov constraints on the object boundaries.

In a subsequent paper Nahi and Lopez-Mora (32) use a more complex  $\gamma$  function. For each row,  $\gamma$  either indicates the absence of an object or provides a vector estimate of the object's width and geometric center in that row. Thus, the two-element vector contains information about the object's size and skewness. The vectors corresponding to successive rows are assumed to define a first-order Markov process.

Cooper (33) views the Nahi approach as too restrictive, in that it reduces the original two-dimensional problem to a one-dimensional problem. Cooper investigates the general class of images formed by a blob of constant gray level on a constant background, with additive white Gaussian noise over the entire image. He uses a Markov process to model the blob boundary and constructs a derivative field by taking the directional derivative at each pixel. He then estimates the blob boundary by maximizing the joint likelihood of a hypothetical blob boundary and of all the image data contained in the directional derivative field.

Recursive solutions based on differential (difference) equations are common in one-dimensional signal processing and have been generalized to two dimensions. Jain (34) investigates the applicability of three kinds of random fields to the image-modeling problem, each characterized by a different class of partial differential equations (PDEs): hyperbolic, parabolic, and elliptic. A digital shape is defined by a finite-difference approximation of a PDE. The class of hyperbolic PDEs is shown to provide more general causal models than autoregressive moving-average models. For a given spectral density (or covariance) function, parabolic PDEs can provide causal, semicausal, or even noncausal representations. Elliptic PDEs provide noncausal models that represent two-dimensional discrete Markov fields and can be used to model both isotropic and anisotropic imagery. Jain argues that the PDE model is based on a well-established mathematical theory and, further, that there exists a considerable body of computer software for numerical solutions. The PDE model also obviates the need for spectral factorization, thus eliminating the restriction of a separable covariance function. System-identification techniques may be used for choosing the PDE model for a given class of images. Chellappa (35) derives a convexity properties of the autocorrelation function for the PDE models. The hyperbolic (elliptic) model gives an autocorrelation function that is convex (concave) along both axes. The autocorrelation function for the parabolic model is convex along one axis and concave along the other.

Matheron's (36) regionalized random-variable approach emphasizes pixel properties whose complex mutual correlation reflects the spatial structure. He assumes weak stationarity of the gray-level increments between pixels. The variogram

$$\gamma(d) = E[(\text{gray level at } i - \text{gray level at } j)^2]$$

where  $d = |i - j|$ , is the basic analytic tool. Huijbregts (37) gives numerous examples of regionalized variables: in geology, the ore grade and thickness, gravity, geochemical content; in forestry, the density of trees; in hydrology, the piezometric height; in meteorology, the quantity of dust and water vapor in the atmosphere. He discusses several properties of the variogram and relates them to the spatial structure of the regionalized variables. The variogram of the residuals with respect to the local mean is used for nonhomogeneous fields with locally varying mean.

Pratt and Faugeras (38) and Gagalowicz (39) view texture as the output of a homogeneous spatial filter excited by white, not necessarily Gaussian, noise. A texture is characterized by its mean, the histogram of the input white noise, and the transfer function of the filter. For a given texture the model parameters are obtained as follows:

the mean is readily estimated from the texture;

the autocorrelation function is computed to determine the magnitude of the transfer function; and

higher order moments are computed to determine the phase of the transfer function.

Inverse filtering yields the white-noise image and hence its histogram and probability density. The inverse filtering or decorrelation may be done by simple operators. For example, for a first-order Markov field, decorrelation may be achieved by using a Laplacian operator (38). The whitened field estimate of the independent, identically distributed noise process will only identify the spatial operator in terms of the autocorrelation function, which is not unique. Thus, the white-noise probability density and spatial filter do not, in general, make up a complete set of descriptors (40). (To generate a texture, the procedure can be reversed by generating a white-noise image having the computed statistics and then applying the inverse of the whitening filter.)

A random field may represent variation in gray level, color, elevation, or temperature, among other characteristics. Several researchers have proposed models specifically for height fields. One example is the Longuet-Higgins model (41–43) developed for the ocean surface (see also Ref. 44). Longuet-Higgins treats the ocean surface as a random field satisfying the following assumptions:

the wave spectrum contains a single narrow band of frequencies; and

the wave energy results from a large number of different sources whose phases are random.

The basic equation governing the process is

$$X_{i,j} = \sum_k A_k \cos(u_k i + u_k j + \phi_k)$$

Longuet-Higgins (43) obtains the statistical distribution of wave heights. He also derives the relationships governing the



root-mean-square wave height, the mean height of a given highest percentage of waves, and the most likely height of the largest wave in a given region.

Longuet-Higgins also obtains a set of statistical relations among parameters for a random moving surface (42) and a Gaussian isotropic surface (43). Some of the results he obtains concern the following:

- the probability distribution of the surface elevation;
- the probability distribution of the magnitude and orientation of the gradient;
- the average number of zero crossings per unit distance along an arbitrarily placed line transect;
- the average contour length per unit area;
- the average density of maxima and minima; and
- the probability distribution of the heights of maxima and minima.

All results are expressed in terms of the two-dimensional energy spectrum. He also studies and solves the converse problem: given certain statistical properties of the surface, determine a convergent sequence of approximations to the energy spectrum.

Longuet-Higgins's treatment of images of ocean waves makes use of features and analyses that may be used for image modeling in general. Panda (45) adopts the Longuet-Higgins approach to analyze background regions selected from forward-looking infrared (FLIR) imagery. (A FLIR device uses an array of heat-sensing elements to record an image. Infrared sensors are in wide use by the military, since infrared images can be made at night and do not require transmission of energy to be reflected.) He derives expressions for the density of border points and average number of connected components along a row of a thresholded image and obtains generally good agreement between observed and predicted values. He uses the same approach (46) to predict the properties of images resulting from the application of edge detectors.

Schachter (47) describes a variation of the Longuet-Higgins model. The random field is described by the basic equation

$$X_{i,j} = \mu + \frac{1}{m} \sum_{k=1}^m A_k n(u_k i + u_k j = \phi_k) \quad m \leq 3$$

where  $n(\cdot)$  denotes a narrow-band noise waveform of a given center frequency and bandwidth, and  $m$  denotes the number of waveforms. A narrow-band noise waveform may be thought of as an envelope function modulating a carrier frequency. Thus this model describes textures of a two-level hierarchy; see Figure 2 for examples. The model has been implemented in hard-

ware in a real-time image-generation system for flight simulation.

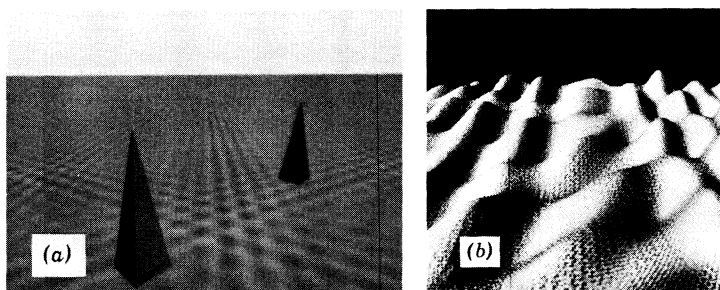
Several authors describe models for the earth's surface. Freiburger and Grenander (48) reason that the earth's surface is too irregular to be represented by an analytic function that has only a small number of free parameters. Nevertheless, because landscapes possess strong continuity properties, they suggest using stochastic processes derived from physical principles. Mandelbrot (49) uses a Poisson-Brown surface to give a first approximation to the earth's relief. The earth's surface is assumed to have been formed by the superimposition of very many, very small cliffs along straight faults. The positions of the faults and the heights of the cliffs are assumed random and independent. Mandelbrot suggests that the generated surface could be made to resemble some actual terrain more closely by introducing anisotropy into ridge directions. Mandelbrot's model is often used in computer graphics to generate artificial terrain scenes. Adler (50-56) presents a theoretical treatment of Brownian sheets and relates them to the rather esoteric mathematical concept of Hausdorff dimension. Mark (57) discusses Brownian sheets and a number of other approaches to modeling the topological randomness of geomorphic surfaces.

**Local Models.** Global models characterize spatial variation in the whole image as a single random field. If neither knowledge nor hypotheses about the type of random field are available, a class of local models is used. Local models assume relationships among gray levels of pixels in small neighborhoods. Predetermined formalisms are used to describe such relationships. The modeling process consists of choosing a formalism and evaluating its parameters. Two basic categories of local models arise from the joint and conditional probability formulations of variation in a neighborhood proposed by Whittle (58) and Bartlett (59,60). Whittle's definition requires that the joint probability distribution of the variables in a given neighborhood be of the product form

$$\prod_{i,j} Q_{i,j}(x_{i,j}, x_{i-1}, x_{i+1}, x_{i,j-1}, x_{i,j+1}, \dots)$$

where  $x_{i,j}$  is a realization of the random variable  $X_{i,j}$  associated with pixel  $(i,j)$  and  $Q$  is a non-negative function. Bartlett's definition requires that the conditional probability distribution of  $X_{i,j}$  depend only on the values at the neighbors of  $(i,j)$ . Besag (61), however, describes a number of difficulties with this approach. There is no obvious way to obtain a joint probability structure for a given conditional probability model; the conditional probability structure is subject to some subtle, yet highly restrictive, consistency conditions, and when these restrictions are enforced, it can be shown (15) that the conditional formulation is degenerate with respect to the joint formulation. Besag (61) found that the constraints on the conditional probability structure are so severe that they actually dictate particular models. (For example, for binary variables the conditional probability formulation gives rise to the Ising model of statistical mechanics.)

The conditional probability approach, however, has served as the basis for a commonly used class of models, called Markov-image models. Consider a finite image of  $n$  pixels with an associated collection of  $n$  random variables  $\{X_i\}$ . Suppose that for each pixel  $i$  the conditional distribution of its associated random variable  $X_i$ , given all other pixel values, depends only on those pixels within a finite neighborhood  $N(i)$  of pixel  $i$ . Pixel  $j$  ( $\neq i$ ) is said to be a Markov neighbor of pixel  $i$  if and only if the functional form  $\Pr(x_i | x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$  is



**Figure 2.** Three intersecting long-crested narrow-band waveforms shown in perspective: (a) as a texture; (b) as a height field.



dependent on the variable  $x_j$ . The complete set of neighborhoods  $N_1, \dots, N_n$  generate an associated class of valid probability distributions for  $(X_1, \dots, X_n)$ . Any member of this class is formally called a Markov (random) field. The use of the term "Markov" in a two-dimensional context is somewhat controversial. Wong (62) e.g., offers a proof that there exists no continuous two-dimensional random field that is both homogeneous and Markov. Serra (63) also discusses the misuse of the term "Markov" in digital image modeling. One result of the important Clifford-Hammersley theorem is that for any Markov field, the functional form

$$\Pr(X_i = x_i, X_j = x_j, \dots, X_s = x_s \mid \text{all other pixel values})$$

depends only on  $x_i, x_j, \dots, x_s$  and the values at neighboring sites of  $i, j, \dots, s$ .

Different choices for the set of a pixel's neighbors give rise to different Markov models. A first-order (or nearest neighbor) Markov model lets the value at each pixel depend only on the values at its 4-neighbors (two horizontal and two vertical neighbors). A second-order scheme uses an 8-adjacent neighborhood (the  $3 \times 3$  neighborhood centered at the pixel), etc. The  $m$ th-order Markov neighborhood of pixel  $i$  is denoted by  $N^m(i)$ . Second-order models are much more complex than first-order models, and unless the variables are assumed to be Gaussian, third- and higher order schemes are too cumbersome to exploit. First- and second-order schemes may be extended to other lattice structures (61), nonlattice structures (64), higher dimensions, and the space-time domain (65), as well.

The specification of the probability distribution of a pixel's gray level, given the gray levels of its neighbors, defines a strict-sense Markov field representation (19). Besag describes two of the more interesting models of this kind (61), the autobinary and the autonormal schemes. The autobinary scheme has a conditional structure of the form

$$\Pr(x_i \mid x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = \frac{\exp[x_i(\alpha_i + \sum_{j \in N(i)} \beta_{i,j} x_j)]}{1 + \exp[\alpha_i + \sum_{j \in N(i)} \beta_{i,j} x_j]}$$

where  $x_i$  can take on only values of 0 or 1.

The autonormal scheme assumes that the joint distribution of the variables is multivariate normal. For this case, the conditional probability function is given by

$$\Pr(x_i \mid x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = (2\pi\sigma^2)^{-1/2} \exp[-\frac{1}{2}\sigma^{-2}\{(x_i - \mu_i)^t \sum_{j \in N(i)} \beta_{i,j}(x_j - \mu_j)\}^2]$$

The joint density function is given by

$$\Pr(\mathbf{x}) = (2\pi\sigma^2)^{-n/2} |B|^{1/2} \exp[(\mathbf{x} - \mu)^t B(\mathbf{x} - \mu)/2\sigma^2]$$

where  $\mathbf{x} = (x_1, \dots, x_n)$ ,  $\mu = E[\mathbf{X}]$ , and  $B$  is a matrix defined as

$$b_{i,j} = b_{j,i} = \begin{cases} 1 & \text{if } i = j \\ -\beta_{i,j} & \text{if } j \in N(i) \text{ } j \neq i \\ 0 & \text{otherwise} \end{cases}$$

Also,  $E[X_i \mid \text{all other pixel values}] = \mu_i + \sum_{j \in N(i)} \beta_{i,j}(x_j - \mu_j)$ . For a first-order autonormal scheme,  $X_{i,j}$  (given the values at all other pixels) is normally distributed with mean

$$\alpha + \beta_1(x_{i-1,j} + x_{i+1,j}) + \beta_2(x_{i,j-1} + x_{i,j+1})$$

and common variance  $\sigma^2$ .

For a discussion of other autoschemes, including the autobinomial, auto-Poisson, and autoexponential schemes, see Besag (61). Cross and Jain (66) report experiments on fitting the autobinomial model to textures from Brodatz (67). Maximum-likelihood estimates of the model parameters are used to test the hypothesis that a given texture sample is described by an autobinomial model with the estimated parameters. The model is also used to generate texture samples.

A close relative of the automodel is the simultaneous autoregressive scheme, defined by

$$X_i = \mu_i + \sum_{j \in N(i)} \beta_{i,j}(X_j - \mu_j) + e_j$$

where error (noise) terms are independent Gaussian variables with zero mean and common variance  $\sigma^2$ . The joint probability density is given by

$$\Pr(\mathbf{X}) = (2\pi\sigma^2)^{-n/2} |B| \times \exp[-2\sigma^{-2}(\mathbf{X} - \mu)^t B(\mathbf{X} - \mu)]$$

where  $B$  is a nonsingular matrix defined as

$$b_{i,j} = \begin{cases} 0 & \text{if } i = j, \\ -\beta_{i,j} & \text{if } j \in N(i) \text{ } j \neq i, \\ 0 & \text{otherwise} \end{cases}$$

A first-order scheme of this type is given by

$$X_{i,j} = \beta_1 X_{i-1,j} + \beta_2 X_{i+1,j} + \beta_3 X_{i,j-1} + \beta_4 X_{i,j+1} + e_{i,j}.$$

For this simple model

$$\begin{aligned} E[X_{i,j} \mid \text{all other pixel values}] &= \alpha[(\beta_1 + \beta_2)(x_{i-1,j} + x_{i+1,j}) \\ &\quad + (\beta_3 + \beta_4)(x_{i,j-1} + x_{i,j+1}) \\ &\quad - (\beta_1\beta_4 + \beta_2\beta_3)(x_{i-1,j-1} + x_{i+1,j+1}) \\ &\quad - (\beta_1\beta_3 + \beta_2\beta_4)(x_{i-1,j+1} + x_{i+1,j-1}) \\ &\quad - \beta_3\beta_4(x_{i,j-2} + x_{i,j+2}) \\ &\quad - \beta_1\beta_2(x_{i-2,j} + x_{i+2,j})] \end{aligned}$$

where  $\alpha + 1/(1 + \beta_1^2 + \beta_2^2 + \beta_3^2 + \beta_4^2)$  (61) and  $\text{Var}(X_{i,j} \mid \text{all other pixel values}) = \alpha\sigma^2$ .

Besag (64) notes that the symmetry requirements of the simultaneous autoregressive model (i.e.,  $\beta_1 = \beta_2$ ;  $\beta_3 = \beta_4$ ) are automatically fulfilled without the need to place prior restrictions on them. He also notes that  $e_{i,j}$  and  $x_{i',j'}$  are uncorrelated whenever  $(i,j) \neq (i',j')$ .

Techniques for estimating the unknown parameters of these models have not been described. Good discussions of this topic can be found in Refs. 8, 61, and 68-71. The relationship between causal and noncausal Markov neighborhoods has also been studied in two dimensions. Abend et al. (72) use Markov chain methods to show that in many cases a noncausal dependence is equivalent to a causal dependence. Woods's (73) treatment allows a larger number of equivalent pairs of causal and noncausal neighborhoods. And still other studies present additional examples and discussion of Markov models (24,74-80).

Kashyap (81) uses circulant matrices to define autoregressive models for an infinite array (the array obtained by infinitely repeating a given finite image). This obviates the need for the initial conditions required by the nonperiodic autoregressive processes. Kashyap obtains the probability density of the data and discusses the maximum-likelihood estimation of the model parameters. He then gives decision rules for the choice of neighbors of a pixel (see also Ref. 82) and for testing the homogeneity of image data. For nonhomogeneous images

he uses multivariate autoregressive models that are constructed as follows. An image is divided into small blocks each of which is assumed to be homogeneous; a univariate autoregressive model is then obtained for each of the windows, and the parameter vectors of these models are then viewed as data and are modeled by a multivariate random field. Finally, Kashyap (81) obtains an expression for the probability density of the field.

Links and Biemond (83) (see also Refs. 84 and 85) have investigated the autocorrelation separability of image models. They consider a general model of the form

$$X_i = \sum_{j \in N(i)} \beta_{i,j} X_j + \gamma e_j$$

or in vector-matrix notation

$$\underset{n \times 1}{\mathbf{X}} = \underset{n \times n \times 1}{B\mathbf{X}} + \underset{n \times 1}{\gamma\mathbf{e}}$$

where  $n = M^2$ . The model autocorrelation is clearly

$$E[\mathbf{X}\mathbf{X}^t] = \gamma^2 B^{-1} E[\mathbf{e}\mathbf{e}^t] (B^{-1})^t$$

The  $n \times n$  autocorrelation matrix  $E[\mathbf{X}\mathbf{X}^t]$  may be separable into a direct product of an  $M \times M$  column autocorrelation matrix  $B_c$  and an  $M \times M$  row autocorrelation matrix  $B_r$ ; i.e.,

$$\begin{aligned} E[\mathbf{X}\mathbf{X}^t] &= E[\mathbf{X}_c \mathbf{X}_c^t] \otimes E[\mathbf{X}_r \mathbf{X}_r^t] \\ &= \gamma^2 (B_c^{-1} \otimes B_r^{-1}) E[\mathbf{e}_c \mathbf{e}_c^t] \otimes E[\mathbf{e}_r \mathbf{e}_r^t] (B_c^{-1} \otimes B_r^{-1})^t \end{aligned}$$

where  $\mathbf{X}_r = (X_{i,1}, \dots, X_{i,M})$ ,  $\mathbf{X}_c = (X_{i,j}, \dots, X_{M,j})^t$ .

Thus, for separability of the model autocorrelation, it is only necessary that the model operator be separable into a column operator  $B_c$  and a row operator  $B_r$  and that the error autocorrelation  $E[\mathbf{e}\mathbf{e}^t]$  be separable into a column autocorrelation  $E[\mathbf{e}_c \mathbf{e}_c^t]$  and a row autocorrelation  $E[\mathbf{e}_r \mathbf{e}_r^t]$ .

If the elements of  $\mathbf{e}$  are uncorrelated variables with zero mean and variance  $\sigma^2$ , the autocorrelation matrix  $E[\mathbf{e}\mathbf{e}^t]$  is separable: i.e.,  $E[\mathbf{e}\mathbf{e}^t] = \sigma^2 I_c \otimes I_r$ . The  $\beta_{i,j}$  terms can be written in a matrix of the form

$$\begin{array}{ccccccc} & & & & & & \\ & & & & & & \\ & & & & & & \\ \cdots & \beta_{-1,-1} & \beta_{-1,0} & \beta_{-1,1} & \cdots & & \\ \cdots & \beta_{0,-1} & -1 & \beta_{0,1} & \cdots & & \\ \cdots & \beta_{1,-1} & \beta_{1,0} & \beta_{1,1} & \cdots & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{array}$$

If every two rows of the matrix are pairwise linearly independent, the model autocorrelation is separable.

The local models discussed so far relate the gray level of a pixel to the gray levels of its neighbors using a conditional probability formulation. The joint probability approach has also received considerable attention. Its primary difficulty, however, is the high dimensionality of the joint probability densities, even for small neighborhoods, making parameter estimation complex and cumbersome. Read and Jayaramamurthy (86) and McCormick and Jayaramamurthy (87) make use of switching-theory techniques to reduce this intractability. They develop minimal functions for modeling local gray-level patterns as follows. Suppose that each pixel is assigned one of  $N$  gray levels; then a neighborhood of  $M$  pixels may be represented by a point in an  $(M \times N)$ -dimensional space. Points corresponding to an aggregate of neighborhoods of a

single pattern are likely to form clusters in this space. A generalization of standard switching theory, the set-covering methodology of Michalski and McCormick (88) is used to describe the set of points in a cluster. These maximal descriptions allow coverage of empty spaces in and around clusters. The sample size need only be large enough to provide a reasonable representation. This approach has the advantage of a simple table look-up decision for classifying textures (89).

A number of investigators confine joint statistics to neighborhoods of size two. A texture is characterized in terms of its gray-level cooccurrence tallies, which are the first estimate of the corresponding joint probabilities. Rosenfeld and Troy (90) and Haralick (89,91-93) suggest the use of two-dimensional spatial dependence of gray levels for fixed distances and angular separations. Haralick and numerous other investigators apply features derived from the cooccurrence matrix to various texture-classification and discrimination problems. The performance of these features as texture measures is compared with several other techniques by Weszka, Dyer, and Rosenfeld (94) and Connors and Harlow (95). Zucker and Terzopoulos (96) note that if a cooccurrence matrix is treated as a contingency table, a standard  $\chi^2$  test can be used to see whether the rows and columns of the matrix are independent. (Presumably, any dependence found here is due to the structure in the image.) They use the maximum among  $\chi^2$  values corresponding to various choices of distance and orientation to identify structural characteristics of the image. Their particular approach has been used in geology for many years (97-99), although the classes are separate entities (mineral types), not ordered gray levels. Davis et al. (100) suggest the use of a generalized cooccurrence matrix based on local features rather than gray levels.

**Syntactic Models.** Rosenfeld (101) defines a procedural model as "any process that generates or recognizes images; the class that it defines consists of the images that it accepts." Grammatical (or syntactic) models fall into this category.

Conventionally, a language is defined as a set of strings over an alphabet, where the alphabet consists of the set of all symbols which can appear in the strings of the language, and a string is a finite ordered sequence of symbols. A grammar is a set of rules which define how the strings of the language are formed. Equivalently, a grammar can be used to recognize the language's strings by using the rules in reverse order. This concept can be generalized in a number of ways (102-105) to define grammars for classes of images.

An array grammar generates images by repeatedly replacing subarrays by other subarrays. A stochastic array grammar is one in which the replacement rules are probabilistic.

In one-dimensional grammars there is no problem with replacing one string by another. But for array grammars the shapes of subarrays must be compatible so that there are no "holes" in the resulting image. Jayaramamurthy (106) attempts to solve this problem with multilevel grammars. Here the subarrays at levels greater than zero are patterns of a specified shape, with the patterns themselves derived by a set of grammars at the next lower level. All of the terminal subarrays at any level represent patterns of the same shape.

Syntactic methods have been used for locating highways and rivers in LANDSAT images and for texture modeling [see Fu (107) for additional references]. Although the use of subarrays in the vocabulary of array grammars poses many difficulties, it gives these models a flavor of the second major class of models, the region-based models.

### Region-Based Models

These models view an image as a set of subpatterns placed according to a given set of rules. Both the subpatterns and their arrangement may be defined statistically, and these subpatterns themselves may be hierarchically composed of smaller patterns. Again, as with pixel-based models, the objective is to model a single texture; the concept of regions is used only to capture the microstructure.

Region-based models are defined using regions, instead of pixels, as primitives. A given model specifies the shapes of the regions and gives the rules for their placements in the plane, thereby allowing increased control over some pattern characteristics. Both the shapes and the placement rules may be specified statistically.

Over the years, region-based models have received far less attention than the pixel-based models in computer image processing. But recently these models have been investigated for texture analysis and synthesis. One class studied is that of mosaic models. These models view an image as a mosaic constructed by tessellating the plane into cells and then coloring the cells by some given process. The resultant pattern consists of color patches, each patch formed by identically colored contiguous cells. Tessellations commonly used include the regular triangular, square, and hexagonal tessellations and random tessellations such as the following:

*Poisson line:* A Poisson process chooses pairs  $(\rho, \theta)$ ,  $0 \leq \theta \leq \pi$ ,  $-\infty < \rho < \infty$ . The lines  $x \cos \theta + y \sin \theta = \rho$  define a tessellation of the plane.

*Voronoi:* A Poisson process chooses points (nuclei) in the plane. Each nucleus defines a "Dirichlet cell" consisting of

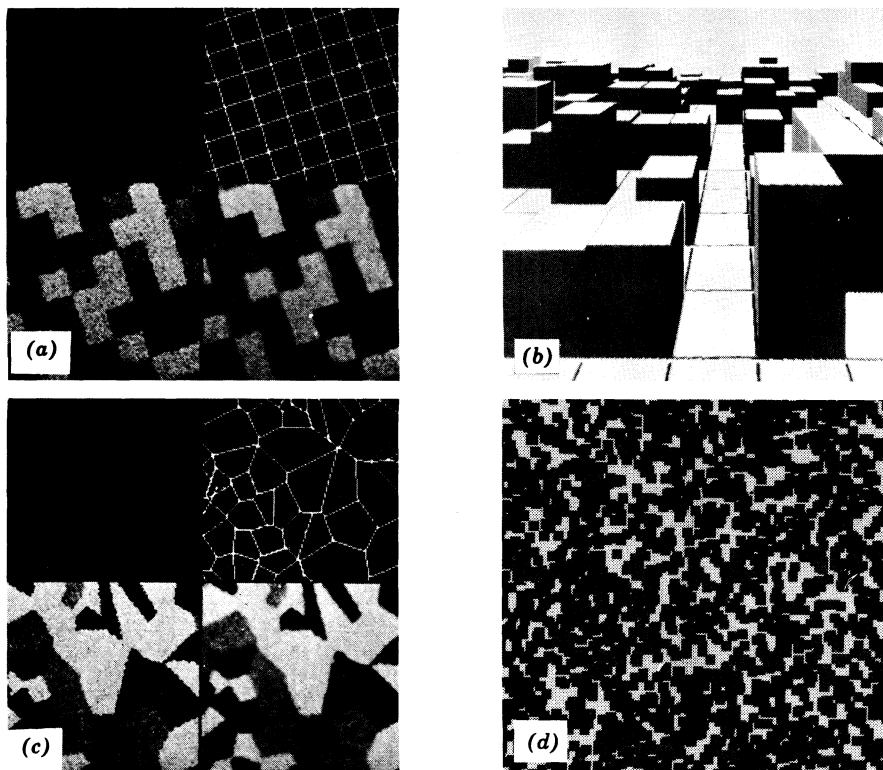
all the points in the plane nearer to it than to any other nucleus.

*Delaunay:* All pairs of nuclei whose Dirichlet cells are adjacent are joined by straight-line segments to define the tessellation.

The coloring process most commonly used assigns one of a predetermined set of colors to a cell, according to a given probability vector. (The term "color" may stand for a fixed gray level or for gray levels from a given distribution of a vector of red, green, and blue components, among other possibilities.)

Another class of region-based models consists of the coverage (or "bombing") models. These models view an image as a random arrangement of a given set of colored shapes over a uniform background. Once again, the choice of shapes and placement rules specifies a particular model. Circles have often been used for the figures, placed at locations chosen by a Poisson process. Figure 3 shows some examples of random mosaics and coverage patterns.

Region-based models of the above types, mosaic and coverage, have been popular in many disciplines, including geology, forestry, biology, ecology, astronomy, crystallography, and statistics. Several properties of mosaics and coverage patterns have been obtained by researchers in these fields. Many of the point correlation properties are discussed in Matern's (108) excellent thesis on spatial variation. Switzer (109–111) and Pielou (112) extend Matern's results. They (109,112) show that a Poisson line mosaic formed from independently colored cells has the interesting property that any sequence of colors along a transect forms a Markov chain. The Markov properties of the Poisson line model have also been investigated by Scheaffer



**Figure 3.** Examples of random mosaics and coverage patterns (a) Square tessellation and mosaic. (b) Square mosaic displayed as a height field in perspective. (c) Voronoi tessellation and mosaic. (d) Square coverage pattern.

(113,114). Switzer (109) derives an expression for the autocorrelation function of the Poisson line mosaic, and Matern (115,116) discusses some geometric properties of the cells in random tessellations. Extensive work on estimating the geometric characteristics of cells in Poisson line tessellations has been done by Miles (117,118), Crain and Miles (119), and Richards (120). They estimate properties such as the expected area of a cell, the expected number of sides of a cell, the expected perimeter of a cell, and the expected number of cells meeting at a vertex. Modestino et al. (121–123) discuss joint pixel properties for mosaics with correlated cell colors. Zucker (124) views texture as a distortion of an ideal mosaic. The distortion is specified in terms of geometric transforms applied to the mosaic.

Ahuja (125–127) derives extensive results on the properties of connected color components in regular mosaics (triangular, square, and hexagonal) and in random mosaics (Poisson line, Voronoi, and Delaunay). Among the properties analyzed are the expected component area, the component perimeter, the component width, the component density, and the point correlation properties. The Voronoi mosaic is of special interest because its cells mimic those formed by natural-growth processes. Matern (108) gives its autocorrelation function as a double-integral equation having no elementary solution. Ahuja (125) and Moore (128) provide empirical estimates. The geometric characteristics of the Voronoi tessellation are covered by Miles (129), Gilbert (130), and Lee (131). Santalo (132) is a good reference for the properties of various tessellations.

Solomon (133) first popularized the circular coverage model. Analogous to the analysis of mosaics, Ahuja (125,127,134) obtained properties of the color components and joint characteristics of pairs of points for coverage patterns. Moore (135,136) has investigated anisotropic random mosaics. The application of region-based models to modeling images has received very little attention. Connors and Harlow (95) start with a real-world texture and attempt to find a set of unit patterns and placement rules that could generate the texture. Ahuja et al. (137) and Schachter et al. (138) attempt fitting mosaic models to a variety of textures. Both of them choose the model and its parameters so as to provide a match between the observed and expected values of component properties. Modestino et al. (122) report experiments with texture discrimination using mosaic models. A detailed treatment of many aspects of mosaic and coverage models appeared in Ref. 139.

Region-based models are further discussed by Rosenfeld and Lipkin (140), Serra et al. (63, 141), Miles (142), Matheron (143), Ripley (144), Grunbaum and Shepard (145), Haralick (2), Schachter and Ahuja (146), Ahuja and Rosenfeld (147), and Davis and Mitiche (148). Syntactic models mentioned herein have also been used as region-based models with regions instead of pixels defining the terminal symbols.

### Models for Human Texture Perception

Most of the work described above uses models and features of textures that are selected based on intuition and, at best, have limited generalizability. Thus, different textures require distinct methods of analysis. Since the success of texture recognition, discrimination, or synthesis is measured by one's judgment and evaluation, the goal of texture modeling is to achieve human-level performance. The question then arises as to which properties are given importance by the human visual system and which features are ignored. Julesz (149–151) has

done extensive work on identifying characteristics of textures important in texture perception by humans. Julesz's first conjecture stated that the human visual system cannot perceive differences in textures that are identical in second-order gray-level statistics. Gagalowicz (152) suggested that only local second-order statistics may be used for human texture discrimination. Julesz later modified his conjecture to state that the human visual system can only perceive (first-order) statistics of texture features called textons. Color, line terminators, and edge segments are examples of textons. Thus, two textures are perceived to be different by humans if they differ in the mean number per unit area of edge segments of a given orientation, end points of edge segments, or sharp corners of regions or if the colors of texture elements are different. Julesz has demonstrated the role of textons by synthesizing textures with controlled statistical properties and testing pairs of textures with known properties for discriminability by humans.

Mandelbrot (153) has proposed the use of sets having fractional dimension, or fractal sets, to model textures. A fractal set is defined as a set for which the Hausdorff–Besicovitch dimension strictly exceeds the topological dimension. Intuitively, fractal dimension characterizes the perceived roughness of an image texture of the corresponding three-dimensional surface. Pentland (154) reports experiments that demonstrate a high correlation between perceived roughness and fractal dimension of textures. Pentland has used homogeneity of fractal dimension as a criterion to perform image segmentation.

### Inference of Three-Dimensional Shape from Texture

The third important application of texture analysis is in inference of three-dimensional structure of a scene from images. Perception of scene properties depends on a variety of cues present in the images. Texture variation represents one such cue. A uniformly textured surface undergoes two types of distortions during the imaging process:

A uniform compression of increasingly large areas of surface onto a fixed area of the image occurs as the textured surface recedes from the viewer.

An anisotropic compression of the texture elements due to foreshortening occurs as the surface tilts away from the frontal plane.

The resulting texture gradients provide information about the relative distances and orientations of the textured surfaces visible in the image. As the distance to a texel increases, the visual angle subtended by the texel decreases. A more distant texel occupies a smaller area of the image. This influence of distance on perceived texel extent is isotropic: All dimensions of the texel are scaled equally as distance changes.

The effect of foreshortening on apparent texel size and shape is anisotropic: Some dimensions of the texture element are shrunk more than others. Foreshortening is a compression of the texture in the direction of the tilt. The amount of compression is proportional to slant.

Since even homogeneous texture modeling is a challenging task, to extract surface-shape characteristics from textural nonhomogeneity is a difficult problem. To solve the problem requires separation of systematic projective distortion from unsystematic spatial variance in attributes of texels such as

their locations, shapes, and gray-level characteristics. The latter variance represents the randomness inherent in the notion of texture. The hierarchical nature of textures further confounds the problem since in images showing large perspective depth changes, the nearby regions show greater detail and hence finer texture that may be very different from the coarser scale texture visible in distant regions.

Work on inferring three-dimensional scene characteristics from image texture is more recent, limited, and more exploratory than the work described in earlier sections. No significant understanding of the inference process exists, nor are there available any approaches that even begin to tackle the problem in any generality.

In one of the first attempts at formulating this problem Bajcsy and Lieberman (155) use Fourier domain features to characterize texture coarseness and elongatedness (see also Multisensor integration). Texture gradients are detected by calculating Fourier transforms of various parts of the image, determining a characteristic texture-element size from peaks in the Fourier power spectrum and looking for trends of the characteristic sizes across the image.

Stevens discussed methods of separating these two effects. He proposed to identify the nonforeshortened dimension of each texel, which he called the characteristic dimension (CD) (156). The CD is perpendicular to the gradient of the distance from the viewer and hence perpendicular to the tilt direction. The length of the CD depends only on the distance to the texel and is independent of the slant and tilt. Successful identification of the CDs everywhere in the image provides two pieces of information: relative distance and the tilt direction. A series of relative-distance measures can be differentiated to obtain the surface orientation indirectly. The aspect ratio of the texels provides a measure of the amount of foreshortening. However, the relationship between the aspect ratio and the slant depends on properties of the texture. Stevens presents a good theoretical discussion of the problems involved in defining appropriate texture measures for the extraction of distance and/or surface-orientation information. However, he offers only sketchy suggestions for implementation.

Kender (157) discusses shape-from-texture algorithms designed to work with man-made textures. Specifically, he deals with aerial views of Pittsburgh. He exploits the presence of texture regularities such as equal-area texels, parallel or perpendicular lines, equal spacing, equal-length lines, and symmetry. Kender's main paradigm may be summarized as follows.

Identify some textural property to "regularize." This property is assumed to be more regular in a frontal view of the texture than in the image. For example, nearly parallel lines in the image may be assumed to originate from precisely parallel lines on the surface.

Divide the image into significant subimages.

For each subimage compute all possible back projections. Choose the surface orientation that has the most regularized back projection. (A "back projection" effectively inverts the foreshortening transformation.)

Witkin (158) proposes a simple method for estimating surface orientation in orthographic images of natural textures. He assumes that any systematic elongation in a texture is due to foreshortening and calculates the deprojection that best re-

moves the systematic elongation. The elongation that is present in the image is calculated as follows: apply an edge detector; count the number of edge elements that occur at each possible edge orientation; and calculate which surface orientation would best account for peaks in the edge-orientation histogram (e.g., a preponderance of horizontal edge segments suggests that the surface has been rotated around the  $x$  axis). Witkin's idea is appealing in its simplicity. However, some restrictions apply. The image is assumed to be an orthographic projection, so that there are no distortions due to increasing distance from the viewer. Witkin's method fails for elongated textures such as grass, hair, waves, or straited rock: his method attempts to attribute all of the elongatedness to foreshortening, thereby grossly overestimating the slant. Dunn (159) describes a series of experiments with implementations of three variations of Witkin's algorithm.

### Concluding Remarks

Most work on texture analysis is on modeling texture as two-dimensional, gray-level variation. The choice of models and features here is usually based on intuition. Studies have been done to evaluate the significance of various textural features in human texture perception and characterization of texture perception capabilities of the human visual system. Most recently computational methods have been examined to infer three-dimensional shape of a homogeneously textured surface from spatial variation of corresponding image texture. This is the area that is the least understood of the three, and needs much further work if texture analysis is to play a role in computer vision (qv) anywhere near as powerful as in visual perception in humans.

### BIBLIOGRAPHY

1. R. M. Pickett, Visual Analysis of Texture in the Detection and Recognition of Objects, in B. S. Lipkin and A. Rosenfeld, (eds.), *Picture Processing and Psychopictorics*, Academic Press, New York, pp. 289-308, 1970.
2. R. M. Haralick, Statistical and Structural Approaches to Texture, in *Proceedings of the Fourth International Joint Conference on Pattern Recognition*, November 1978, pp. 45-69.
3. J. J. Gibson, *The Perception of the Visual World*, Houghton-Mifflin, Boston, MA, 1950.
4. J. J. Gibson, *The Sensors Considered as Perceptual Systems*, Houghton-Mifflin, Boston, MA, 1966.
5. N. Ahuja and B. Schachter, "Image models," *ACM Comput. Surv.* 13(4), 373-397 (December 1981).
6. J. K. Hawkins, Textural Properties for Pattern Recognition, in B. S. Lipkin and A. Rosenfeld (eds.), *Picture Processing and Psychopictorics*, Academic Press, New York, pp. 347-370, 1970.
7. J. L. Muerle, Some Thoughts on Texture Discrimination by Computer, in B. S. Lipkin and A. Rosenfeld (eds.), *Picture Processing and Psychopictorics*, Academic Press, New York, pp. 347-370, 1970.
8. J. E. P. Box and G. M. Jenkins, *Time Series Analysis*, Holden-Day, San Francisco, 1976.
9. B. H. McCormick and S. N. Jayaramamurthy, "Time series model for texture synthesis," *Int. J. Comput. Inf. Sci.* 3, 329-343 (1974).
10. J. T. Tou and Y. S. Chang, An Approach to Texture Pattern Analysis and Recognition, in *Proceedings of the 1976 IEEE Conference on Decision and Control*, December 1976, pp. 398-403.

11. J. T. Tou, D. B. Kao, and Y. S. Chang, Pictorial Texture Analysis and Synthesis, *Proceedings of the Third International Joint Conference on Pattern Recognition*, Coronado, CA, 1976.
12. D. W. Bacon, Seasonal Time Series, Ph.D. Dissertation, Universities of Wisconsin, Madison, 1965.
13. P. Whittle, "On stationary processes in the plane," *Biometrika* **41**, 434–449 (1954).
14. M. S. Bartlett, *The Statistical Analysis of Spatial Pattern*, Wiley, New York, 1975.
15. D. Brook, "On the distinction between the conditional probability and joint probability approaches in the specification of nearest-neighbor systems," *Biometrika* **51**, 481–483 (1964).
16. R. Chellappa and N. Ahuja, Statistical Inference Theory Applied to Image Modelling, Technical Report TR-745, Department of Computer Science, University of Maryland, College Park, March 1979.
17. E. Angel and A. K. Jain, "Frame-to-frame restoration of diffusion images," *IEEE Trans. Automat. Contr.* **AC-23** 850–855 (1978).
18. W. K. Pratt, *Digital Image Processing*, Wiley, New York, 1978.
19. A. Rosenfeld and A. C. Kak, *Digital Picture Processing*, Academic Press, New York, 1976.
20. B. R. Frieden, "Statistical models for the image restoration problem," *Comput. Gr. Img. Proc.* **12**(1), 40–59 (January 1980).
21. L. E. Franks, "A model for the random video process," *Bell Syst. Tech. J.* **45**, 609–630 (April 1966).
22. A. Habibi, Two Dimensional Bayesian Estimate of Images, *Proc. IEEE* **60**, 878–883 (July 1972).
23. T. S. Huang, "The subjective effect of two-dimensional pictorial noise," *IEEE Trans. Inform. Theory* **IT-11**, 43–53 (January 1965).
24. A. K. Jain and E. Angel, "Image restoration, modelling, and reduction of dimensionality," *IEEE Trans. Comput.* **C-23**, 470–476 (May 1974).
25. E. R. Kretzmer, "Statistics of television signals," *Bell Syst. Tech. J.* **31**, 751–763 (July 1952).
26. B. R. Hunt and T. M. Cannon, "Nonstationary assumptions for Gaussian models of images," *IEEE Trans. Syst. Man, Cybern.* **SMC-6**, 876–882 (December 1976).
27. B. R. Hunt, "Bayesian methods in nonlinear digital image restoration," *IEEE Trans. Comput.* **C-26**, 219–229 (March 1977).
28. B. R. Hunt, "Nonstationary statistical image models (and their application to image data compression)," *Comput. Gr. Img. Proc.* **12**(2), 173–186 (February 1980).
29. T. Assefi, *Stochastic Processes, Estimation Theory, and Image Enhancement*, Jet Propulsion Laboratory, Pasadena, CA, June 1978.
30. H. J. Trussel and R. P. Kruger, "Comments on 'nonstationary' assumption for Gaussian models in images," *IEEE Trans. Syst. Man Cybern.* **SMC-8**, 579–582 (1978).
31. N. E. Nahi and M. H. Jananshahi, "Image boundary estimation," *IEEE Trans. Comput.* **C-26**, 772–781 (August 1977).
32. N. E. Nahi and S. Lopez-Mora, "Estimation detection of object boundaries in noisy images," *IEEE Trans. Autom. Contr.* **AC-23**, 834–845 (October 1978).
33. D. Cooper, "Maximum likelihood of Markov-process blob boundaries in noisy images," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-1**(4), 372–384 (October 1979).
34. A. K. Jain, "Partial differential equations and finite-difference methods in image processing, Part 1: Image representation," *J. Optimiz. Theory Appl.* **23**, 65–91 (1977).
35. R. Chellappa, On the Correlation Structure of Nearest Neighbor Random Field Models of Images, Technical Report TR-912, Department of Computer Science, University of Maryland, College Park, July 1980.
36. G. Matheron, "The theory of regionalized variables and its application," *Cah. Ctr. Morphol. Math. Font.* **5**, (1971).
37. C. Huijbregts, Regionalized Variables and Quantitative Analysis of Spatial Data, in J. Davis and M. McCullagh (eds.), *Display and Analysis of Spatial Data*, Wiley, New York, pp. 38–51, 1975.
38. W. K. Pratt and O. D. Faugeras, Development and Evaluation of Stochastic-Based Visual Texture Features, in *Proceedings of the Fourth International Joint Conference on Pattern Recognition*, November 1978, pp. 545–548.
39. A. Gagalowicz, Analysis of Texture Using a Stochastic Model, in *Proceedings of the Fourth International Joint Conference on Pattern Recognition*, November 1978, pp. 541–544.
40. W. K. Pratt, O. D. Faugeras, and A. Gagalowicz, "Visual discrimination of stochastic texture features," *IEEE Trans. Syst., Man, Cybern.* **SMC-8**, 796–804 (1978).
41. M. S. Longuet-Higgins, "On the statistical distribution of the heights of sea waves," *J. Mar. Res.* **11**, 245–266 (1952).
42. M. S. Longuet-Higgins, "The statistical analysis of a random moving surface," *Phil. Trans. Roy. Soc. Lond.* **A249**, 321–387 (February 1957).
43. M. S. Longuet-Higgins, "Statistical properties of an isotropic random surface," *Phil. Trans. Roy. Soc. Lond.* **A250**, 151–171 (October 1957).
44. W. J. Pierson, A Unified Mathematical Theory for the Analysis, Propagation and Refraction of Storm Generated Surface Waves, Department of Meteorology, New York University, New York, 1952.
45. D. P. Panda, "Statistical properties of thresholded images," *Comput. Gr. Img. Proc.* **8**, 334–354 (1978).
46. D. P. Panda and T. Dubitzki, "Statistical analysis of some edge operators," *Comput. Gr. Img. Proc.* **9**, 313–348 (December 1979).
47. B. J. Schachter, "Long crested wave models for Gaussian fields," *Comput. Gr. Img. Proc.* **12**, 187–201 (February 1980).
48. W. Freiburger and U. Grenander, Surface Patterns in Theoretical Geography, Report 41, Department of Applied Mathematics, Brown University, Providence, RI, September 1976.
49. B. Mandelbrot, *Fractals—Form, Chance, and Dimension*, W. H. Freeman, San Francisco, CA, 1977.
50. R. J. Adler, "On generalizing the notion of upcrossing to random fields," *Adv. Appl. Prob.* **8**, 798–805 (1976).
51. R. J. Adler and A. M. Hasofffer, "Level crossings for random fields," *Ann. Prob.* **4**(1), 1–12 (1976).
52. R. J. Adler, "A spectral moment estimation problem in two dimensions," *Biometrika* **64**(2), 367–373 (1977).
53. R. J. Adler, "Hausdorff dimension and Gaussian fields," *Ann. Prob.* **5**(1), 145–151 (1977).
54. R. J. Adler, "Some erratic patterns generated by the planar Wiener process," *Suppl. Adv. Appl. Prob.* **10**, 22–27 (1978).
55. R. J. Adler, "Distribution results for the occupation measures of continuous Gaussian fields," *Stoch. Proc. Appl.* **7**, 299–310 (1978).
56. R. J. Adler, "The uniform dimension of the level set of a Brownian sheet," *Ann. Prob.* **6**, 309–315 (1978).
57. D. M. Mark, "Topological Randomness of Geomorphic Surfaces," Technical Report 15, Department of Geography, Simon Fraser University, April 1977.
58. R. Whittle, "Stochastic processes in several dimensions," *Bull. Inst. Int. Stat.* **40**, 974–994 (1963).
59. M. S. Bartlett, *An Introduction to Stochastic Processes*, University Press, Cambridge, UK, 1955.



60. M. S. Bartlett, "Inference and Stochastic Processes," *J. Roy. Stat. A* **130**, 457-477 (1967).
61. J. Besag, "Spatial interaction and the statistical analysis of lattice systems," *J. Roy. Stat. Soc. Ser. B* **36**(2), 192-236 (1974).
62. E. Wong, "Two-dimensional random fields and representations of images," *SIAM J. Appl. Math.* **16**, 756-770 (1968).
63. J. Serra, "The Boolean model and random sets," *Comput. Gr. Img. Proc.* **12**(2), 99-126, (February 1968).
64. J. Besag, "Statistical analysis of nonlattice data," *Statistician* **24**(3), 179-195 (1975).
65. J. Besag, On Spatial-Temporal Models and Markov Fields, *Trans. 7th Prague Conf. Information Theory, Statistical Decision Function, etc.*, Vol. A, Prague, Czechoslovakia, August 1974, pp. 47-55.
66. G. R. Cross and A. K. Jain, "Markov random field texture models," in *Proceedings of IEEE Pattern Recognition and Image Processing Conference*, Dallas, TX, August, 1981.
67. P. Brodatz, *Textures: A Photographic Album for Artists and Designers*, Dover, New York, 1966.
68. D. H. Ballard, C. M. Brown, and J. A. Feldman, An Approach to Knowledge Directed Image Analysis, Technical Report 21, Department of Computer Science University of Rochester, Rochester, NY, 1977.
69. J. Besag, Parameter Estimation for Markov Fields, Technical Report 108, Series 2, Department of Statistics, Stanford University, Stanford, CA, February 1976.
70. J. Besag, "Errors-in-variables estimation for Gaussian lattice schemes," *J. Roy. Stat. Soc. Ser. B* **39**(1), 73-78 (1977).
71. J. K. Ord, "Estimation methods for models of spatial interaction," *J. Am. Stat. Soc.* (1975).
72. K. Abend, T. J. Harley, and L. N. Kanal, "Classification of binary random patterns," *IEEE Trans. Inform. Theor.* **IT-11**, 538-544 (October 1965).
73. J. W. Woods, "Two-dimensional discrete Markovian fields," *IEEE Trans. Inf. Theory* **IT-18**, 232-240 (1972).
74. K. Deguchi, and I. Morishita, Texture Characterization and Texture-Based Image Partitioning Using Two-Dimensional Linear Estimation Techniques, in *U.S.-Japan Cooperative Science Program Seminar on Image Processing in Remote Sensing*, Washington, D.C., November 1-5, 1976.
75. M. Hassner and J. Sklansky, "Markov random field models of digitized image texture" *Patt. Recog.*, 538-540 (November 1978).
76. A. K. Jain, "A semi-causal model for recursive filtering of two-dimensional images," *IEEE Trans. Comput.* **C-26**, 343-350 (1977).
77. D. P. Panda and A. C. Kak, "Recursive least squares smoothing of noise in images," *IEEE Trans. Acoust. Speech Sig. Proc.* **ASSP-25**, 520-524 (1977).
78. D. K. Pickard, "A curious binary lattice process," *J. Appl. Prob.* **14**, 717-731 (1977).
79. D. J. Strauss, "Clustering on colored lattices," *J. Appl. Prob.* **14**, 135-143 (1977).
80. T. R. Welberry and G. H. Miller, "An approximation to a two-dimensional binary process," *J. Appl. Prob.* **14**, 862-868 (1977).
81. R. L. Kashyap, "Random field models of images," *Comput. Gr. Image Proc.* **12**(3), 257-270 (March 1980).
82. R. Chellappa, R. L. Kashyap, and N. Ahuja, "Decision rules for choice of neighbors in random field models of images," *Comput. Gr. Image Proc.* **15**(4), 30-318 (April 1981).
83. L. H. Links and J. Biemond, "On the nonseparability of image models," *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-1**(4), 409-411 (October 1979).
84. H. C. Andrews and B. R. Hunt, *Digital Image Restoration*, Prentice-Hall, Englewood Cliffs, NJ, 1977.
85. W. K. Pratt, "Vector space formulation of two-dimensional signal processing operations," *Comput. Gr. Image Proc.* **4**(1), 1-24 (March 1975).
86. J. S. Read and S. N. Jayaramamurthy, "Automatic generation of texture feature detectors," *IEEE Trans. Comput.* **C-21**, 803-812 (1972).
87. B. H. McCormick and S. N. Jayaramamurthy, "A decision theory method for the analysis of texture," *Int. J. Comput. Inf. Sc.* **4**, 1-38 (1975).
88. R. S. Michalski and B. H. McCormick, "Interval generalization of switching theory, in *Proceedings of the Third Annual Houston Conference on Computing System Science*, Houston, TX, April 1971, pp. 213-226.
89. R. M. Haralick, "The table look-up rule," *Commun. Stat. Theor. Methods* **12**, 1163-1191 (1976).
90. A. Rosenfeld and E. Troy, Visual Texture Analysis, Technical Report 70-116, Department of Computer Science, University of Maryland, College Park, MD, June 1970.
91. R. M. Haralick, A Texture-Context Feature Extraction Algorithm for Remotely Sensed Imagery," in *Proceedings of the 1971 IEEE Decision and Control Conference*, Gainesville, FL, December 1971, pp. 650-657.
92. R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Syst. Man, Cybern.* **SMC-3**, 610-621 (1973).
93. R. M. Haralick and W. F. Bryant, Documentation of Procedures for Texture/Spatial Pattern Recognition Techniques, Technical Report 278-1, Remote Sensing Lab., University of Kansas Center for Research, Lawrence, KS, April 1976.
94. J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A comparison of texture measures for terrain classification," *IEEE Trans. Syst., Man, Cybern.* **SMC-6**, 269-285 (April 1976).
95. R. W. Connors and C. A. Harlow, "A theoretical comparison of texture algorithms," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-2**(3), 204-222 (May 1980).
96. S. W. Zucker and D. Terzopoulos, "Finding structure in cooccurrence matrices for texture analysis," *Comput. Gr. Image proc.* **12**(3), 286-308 (March 1980).
97. J. Harbaugh and G. Bonham-Carter, *Computer Simulation in Geology*, Wiley, New York, 1970.
98. A. B. Vistelius and A. U. Fass, The Mode of Alternation of Strata in Certain Sedimentary Rock Sections, Academy of Sciences USSR, Earth Science Section, Vol. 164, pp. 629-632, 1965.
99. E. H. T. Whitten and M. F. Dacey, "On the significance of certain Markov features of granite textures," *J. Petrol.* **16**(2), 429-453 (1975).
100. L. S. Davis, S. Johns, and J. K. Aggarwal, "Texture analysis using generalized cooccurrence matrices," *IEEE Trans. Patt. Anal. Mach. Int.* **PAMI-1**(3), 251-258 (July 1979).
101. A. Rosenfeld, Image Pattern Recognition, Technical Report TR-850, Computer Science Center, University of Maryland, College Park, MD, January 1980.
102. K. S. Fu, "Stochastic languages for picture analysis," *Comput. Gr. Image Proc.* **2**(3/4), 433-453 (December 1973).
103. S. Y. Lu and K. S. Fu, "A syntactic approach to texture analysis," *Comput. Gr. Image Proc.* **7**, 303-330 (1978).
104. A. Rosenfeld and D. L. Milgram, Array Automata and Array Grammars 2, Technical Report 171, Department of Computer Science, University of Maryland, College Park, MD, 1971.
105. A. Rosenfeld, *Picture Languages*, Academic Press, New York, 1979.

106. S. N. Jayaramamurthy, "Multilevel grammar for generating texture scenes," *Proceedings of the 1979 IEEE Conference on Pattern Recognition and Image Processing*, pp. 391-398.
107. K. S. Fu, "Syntactic image modelling using stochastic tree grammars," *Comput. Gr. Image Proc.* **12**, 136-152 (February 1980).
108. B. Matern, *Spatial Variation*, Medd Statens Skogsforsknings Institut, Stockholm, Sweden, pp. 1-144, 1960.
109. P. Switzer, "A random set process in the plane with a Markovian property," *Ann. Math. Stat.* **36**(6), 1859-1863 (December 1965).
110. P. Switzer, "Reconstructing patterns from sample data," *Ann. Math. Stat.* **38**, 138-154 (1967).
111. P. Switzer, "Mapping a Geographically Correlated Environment, Technical Report 145, Department of Statistics, Stanford University, Stanford, CA, 1969.
112. E. Pielou, *Mathematical Ecology*, Wiley, New York, 1977.
113. R. L. Scheaffer, "An approximate variance for line intersection counts," *J. Microsc.* **103**(3), 343-349 (April 1975).
114. R. L. Scheaffer, "Variance Approximations for Transects: The Poisson Line Model, Technical Report 86, Department of Statistics, University of Florida, Gainesville, January 1975.
115. B. Matern, Stokastika Modeller for Variation 1 Planet (Stochastic Models for Planar Variation), presented at the Third Nordic Conference on Mathematics and Statistics, Umea, Sweden, June 1969.
116. B. Matern, The Analysis of Ecological Maps as Mosaics, presented at the Advanced Institute of Statistical Ecology Around the World, Pennsylvania State University, University Park, July 7, 1972.
117. R. E. Miles, "Random polygons determined by random lines in the plane," *Proc. Nat. Acad. Sci. USA* **52**, 901-907 (1964).
118. R. E. Miles, "Random polygons determined by random lines in the plane II," *Proc. Nat. Acad. Sci. USA* **52**, 1157-1159 (1964).
119. I. K. Crain and R. E. Miles, "Monte Carlo estimates of the distribution of random polygons determined by random lines in the plane," *J. Stat. Comput. Simul.* **4**, 293-325 (1976).
120. P. I. Richards, "Averages for polygons formed by random lines," *Proc. Nat. Acad. Sci. USA* **52**, 1160-1164 (1964).
121. J. W. Modestino, R. W. Fries, and D. G. Daut, "Generalization of the two-dimensional random checkerboard process," *J. Opt. Soc. Am.* **69**(6), 897-906 (June 1979).
122. J. W. Modestino, R. W. Fries, and A. L. Vickers, Texture Discrimination Based upon an Assumed Stochastic Texture Model, Technical Report 79-3, Rensselaer Polytechnic Institute, Troy, NY, July 1979.
123. J. W. Modestino, R. W. Fries, and A. L. Vickers, "Stochastic models generated by random tessellations of the plane," *Comput. Gr. Image Proc.* **12**(1), 74-98 (January 1980).
124. S. Zucker, "Toward a model of texture," *Comput. Gr. Image Proc.* **5**, 190-202 (1976).
125. N. Ahuja, Mosaic Models for Image Analysis and Synthesis, Ph.D. Dissertation, Department of Computer Science, University of Maryland, College Park, MD, 1979.
126. N. Ahuja, "Mosaic models for images, I: Geometric properties of components in cell structure mosaics," *Inform. Sci.* **23**, 69-104 (March 1981).
127. N. Ahuja, "Mosaic models for images, 3: Spatial correlation in mosaics," *Inform. Sci.* **24**, 43-69 (June 1981).
128. M. Moore, The Transition Probability Function of the Occupancy Model, Technical Report 40, Ecole Polytechnique, Montreal, Canada, October 1978.
129. R. E. Miles, "On the homogeneous planar Poisson point process," *Math. Biosci.* **6**, 85-127 (1970).
130. E. N. Gilbert, "Random subdivisions of space into crystals," *Ann. Math. Stat.* **33**, 958-972 (1962).
131. D. T. Lee, On Finding  $k$  Nearest Neighbors in the Plane, Technical Report UILU ENG-76-2216 Coordinated Science Laboratory, University of Illinois, Urbana, May 1976.
132. L. A. Santalo, *Encyclopedia of Mathematics and its Applications: Integral Geometry and Geometric Probability*, Vol. 1, Addison-Wesley, Reading, MA, 1976.
133. H. R. Solomon, "Distribution of the measure of a random two-dimensional set," *Ann. Math. Stat.* **24**, 650-656 (1953).
134. N. Ahuja, "Mosaic models for images, 2: Geometric properties of components in coverage mosaics," *Inform. Sci.* **23**, 159-200 (April 1981).
135. M. Moore, "Non-stationary random set processes with application to pattern reconstruction," *J. Appl. Prob.* **10**, 857-863 (1973).
136. M. Moore, "Anisotropically random mosaics," *J. Appl. Prob.* **11**, 374-376 (1974).
137. N. Ahuja, T. Dubitzki, and A. Rosenfeld, "Some experiments with mosaic models for textures," *IEEE Trans. Syst. Man, Cyber.* **10**, 744-749 (November 1980).
138. B. J. Schachter, L. S. Davis, and A. Rosenfeld, "Random mosaic models for textures," *IEEE Trans. Syst. Man, Cybern.* **SMC-8**, 694-702 (1978).
139. N. Ahuja and B. J. Schachter, *Pattern Models*, Wiley, New York, 1982.
140. A. Rosenfeld and B. S. Lipkin, Texture Synthesis, in B. S. Lipkin and A. Rosenfeld (eds.), *Picture Processing and Psychopictorics*, Academic Press, New York, pp. 309-322, 1970.
141. J. Serra and G. Verchery, "Mathematical morphology applied to fibre composite materials," *Film Sci. Technol.* **6**, 141-158 (1973).
142. R. E. Miles, "A survey of geometrical probability in the plane with emphasis on stochastic image modelling," *Comput. Gr. Image Proc.* **12**(1), 1-24 (1980).
143. G. Matheron, *Elements pour une Theorie des Milieux Poreux*, Masson, Paris, 1967.
144. B. D. Ripley, Stochastic Geometry and the Analysis of Spatial Pattern, Ph.D. Dissertation, Churchill College, May 1976.
145. B. Grunbaum and G. C. Shepard, *Tilings and Patterns*, W. H. Freeman, San Francisco, 1986.
146. B. J. Schachter and N. Ahuja, "Random pattern generation process," *Comput. Gr. Image Proc.* **10**, 95-114 (1979).
147. N. Ahuja and A. Rosenfeld, "Mosaic models for textures," *IEEE Trans. Pattern Analysis Mach. Int.* **3**, 1-11 (January 1981).
148. L. S. Davis and A. Mitiche, "Edge detection in textures," *Comput. Gr. Image Proc.* **12**, 25-39 (1980).
149. B. Julesz, "Visual pattern discrimination," *IRE Trans. Inform. Theory* **8**(2), 84-92 (February 1962).
150. B. Julesz, "Experiments in the visual perception of texture," *Scie. Am.* **232**, 34-43 (April 1975).
151. B. Julesz, "Textons, the elements of texture perception and their interactions," *Nature* **290**, 91-97 (March 1981).
152. A. Gagalowicz, "A new method for texture field synthesis: Some applications to the study of human vision," *IEEE Trans. Patt. Anal. Mach. Intell.* **6**, 520-533 (September 1981).
153. B. B. Mandelbrot, *The Fractal Geometry of Nature*, W. H. Freeman, San Francisco, 1982.
154. A. P. Pentland, "Fractal-based description of natural scenes," *IEEE Trans. Patt. Anal. Mach. Intell.*, **5**, 661-674 (November 1984).
155. R. Bajcsy and L. Liberman, "Texture gradient as a depth cue," *Comput. Graph. Image Proc.* **5**, 52-67 (1976).
156. K. Stevens, "The information content of texture gradients," *Biol. Cybernet.* **42**, 95-105 (1981).
157. J. R. Kender, Shape from Texture, Ph.D. Thesis, Carnegie-Mel-

Ion University Computer Science Department, CMU-CS-81-102, November 1980.

158. A. P. Witkin, "Recovering surface shape and orientation from texture," *Artif. Intell.* 17, 17-45 (1981).
159. S. Dunn, L. Davis, and H. Hakalathi, Experiments in Recovering Surface Orientation from Texture, University of Maryland, Computer Science Technical Report CS-TR-1399, May 1984.

N. AHUJA  
University of Illinois

The work was supported in part by the Air Force Office of Scientific Research under Grant 82-0317 and by the National Science Foundation under Grant 8352408.

## THEOREM PROVING

Automated theorem proving refers to the study and development of programs that reason logically (see Reasoning). Logical reasoning refers to processes that infer new formulas from existing formulas such that the new formula is always true in any interpretation where the old formulas are true. A simple example of logical reasoning is *modus ponens*, in which one infers the formula  $q$  from the two formulas  $p \rightarrow q$  and  $p$ ; it is easy to verify that any assignment of truth values that makes the two hypotheses true must also assign  $q$  to be true. Another example, this one from first-order logic (qv), is to infer  $F(a)$  from  $\forall xF(x)$  where  $F$  is a first-order formula and  $a$  is a term of the language. Again, it is easy to see that the conclusion is true in any interpretation where the hypothesis is true (for a more complete treatment of first-order logic see below or the entry Predicate logic). As a contrast, examples of nonlogical reasoning include making assumptions (e.g., Tweety is a bird, so assume that Tweety can fly, although he could be a penguin, in which case the conclusion would be false), imprecise reasoning where formulas are evaluated in degrees rather than absolute truth or falsity (e.g., HEIGHT(BILL) = 6 ft, so conclude that BILL is tall, even though tall is relative), and others. See the various entries on reasoning for discussions of these kinds of reasoning. Applications of logical reasoning, as embodied by automated theorem-proving programs, obviously include proving conjectures in mathematics but also include performing inferences in other areas as well, and some theorem-proving programs have become useful tools in various areas of research and development. Several of the more important applications and some of the major achievements of automated theorem provers are described below.

### Logic

To understand automated theorem proving, one must be familiar with logic to some extent. A brief discussion of the elements of first-order logic that are most important to the field is given here. (For a more complete presentation, see Propositional logic and Predicate logic.)

**Notation.** A first-order language consists of a number of different kinds of symbols that are combined to make (well-formed) formulas, wffs. These are variable, constant, function, and predicate symbols; the Boolean connectives (AND, OR, NOT, etc.); the quantifiers  $\forall$  (FORALL) and  $\exists$  (EXISTS); pa-

rentheses; and commas. As will become evident below, the choice of what inscriptions to use for each of the first four kinds of symbols is irrelevant. Such choices are usually made to help the person writing or reading the formula, just as in programming languages. Each function and predicate symbol has an arity, or number of arguments, associated with that symbol. The parentheses and commas are used to improve readability. Function symbols are meant to represent functions over the domains of interest, and predicate symbols are meant to represent true-false relationships (as is evident in the next few paragraphs).

The symbols can be combined to form two kinds of formulas, terms and wffs. Any variable or constant symbol by itself is a term, and if  $f$  is an  $n$ -ary function symbol and  $t_1, \dots, t_n$  are terms, then  $f(t_1, \dots, t_n)$  is also a term. Nothing else is a term. There are several rules for forming wffs. First, if  $P$  is an  $n$ -ary predicate symbol and  $t_1, \dots, t_n$  are terms, then  $P(t_1, \dots, t_n)$  is a wff. Since such formulas are the simplest kinds of wffs and play a special role in most areas of theorem proving, they are given a special name, atom. Next, if  $F$  is a wff, then so is NOT  $F$ . If  $F$  and  $G$  are wffs, then so are  $F$  AND  $G$ ,  $F$  OR  $G$ ,  $F$  IMPLIES  $G$ , etc. Finally, if  $F$  is a wff and  $x$  is a variable, then  $\forall xF$  and  $\exists xF$  are also wffs. The notion of the variable  $x$  being bound to the quantifier in the last two wffs is similar to the notion of the scope of a declaration in a block-structured programming language. Thus, those occurrences of  $x$  in  $F$  that are not in the scope of another quantifier over  $x$  inside of  $F$  are bound by the outer quantifier in  $\forall xF$  or  $\exists xF$ . An occurrence of a variable in a wff that is not bound by any quantifier is said to be free. This is just like the occurrence of an undefined variable in a block-structured programming language. A formula that has no free occurrences of variables is said to be closed.

As an example, consider group theory in mathematics. The properties of groups might be represented using a three-place predicate  $P$  with the intention that  $P(x, y, z)$  stands for  $x * y = z$ . A constant symbol, say  $e$ , might also be included to represent the identity and two function symbols, say  $f$  and  $g$ , to represent times and inverse. Wffs such as the following could then be written to represent statements about group theory.  $\forall xP(x, g(x), e)$  would represent the statement that the product of any element with its inverse is the identity.  $\forall x\forall y\exists zP(x, y, z)$  would represent the statement that every pair of elements has a product. A third wff in this language is  $\exists x(\forall y(P(x, y, e)) \text{ IMPLIES } P(y, y, e))$ , which represents the statement that some element of the group has the property that all of its inverses have order 2. Clearly, the first two of the three statements are true for any group, and the last one is not. The notion of "always being true" is a central issue in logic and theorem proving and is made precise below.

Another example illustrates that logic need not be applied only to mathematics problems. Consider a language with two-place predicate symbols ANC and PAR and some constant symbols such as BILL, JOHN, and MARY. Then the formulas PAR(BILL, MARY), PAR(BILL, JOHN), and PAR(MARY, BILL) could represent statements about who is the parent of whom. (Of course, the first and third statements cannot both be true if the interpretation of PAR is parent.) The formula

$$\forall x\forall y(\text{ANC}(x, y) \text{ Iff } (\text{PAR}(x, y) \text{ OR } \exists z(\text{PAR}(x, z) \text{ AND } \text{ANC}(z, y))))$$

represents that ANC is the ancestor relationship defined in terms of parent. To illustrate the point that the symbols them-

selves have no inherent meaning, this problem could have been formalized using the predicate symbols  $Q$  and  $R$  and constant symbols  $a, b, c, \dots$ , with  $Q(a, c)$ ,  $Q(a, b)$ ,  $Q(c, a)$ , and

$$\forall x \forall y (R(x, y) \text{ Iff } (Q(x, y) \text{ OR } \exists z (Q(x, z) \text{ AND } R(z, y))))$$

in place of the preceding formulas.

In view of the last comment, logic requires a precise definition of truth and falsehood. This is provided by the concept of an interpretation. An interpretation of a first-order language consists of a nonempty domain of objects, say  $D$ , and an assignment to the constant, function, and predicate symbols of values, functions, and relations over  $D$ . Thus, in the group-theory example above, one interpretation might have the integers as the domain, plus and unary minus as the assignment to  $f$  and  $g$ , zero as the assignment to the symbol  $e$ , and the relation that is true when the sum of the first two arguments is equal to the third as the assignment to  $P$ . Another assignment might use the rationals as the domain, etc. Clearly, an interpretation can be used to evaluate any closed wff of the language to either true or false. If the interpretation makes a formula  $F$  true, it is said to be a model of  $F$ . A formula may then be either valid (true in all interpretations), satisfiable (true in at least one interpretation), or unsatisfiable (false in all interpretations). Notice that the particular inscriptions used for constant, function, and predicate symbols do not matter. Thus, the two languages used in the preceding paragraph for genealogy have essentially the same interpretations.

Clearly, for any closed wff  $F$ ,  $F$  is valid iff NOT  $F$  is unsatisfiable. In fact, most automated theorem-proving methods work by showing that the negation of a formula is unsatisfiable rather than showing that the formula itself is valid. To use such a program to prove a conjecture (or solve a problem), one formulates the negation of that conjecture (or denies that the problem has a solution) and hopes the program will find a contradiction. Such procedures are called refutation procedures.

**Special Forms and Other Special Concepts.** Several special forms for wffs play a central role in automated theorem proving. The first of these is prenex normal form (PNF). In this form the quantifiers are all at the front of the formula, and the inside part is in conjunctive normal form (CNF); thus

$$Q_1 x_1 \dots Q_n x_n (C_1 \text{ AND } \dots \text{ AND } C_k),$$

where each  $Q_i$  is one of the two quantifiers and each  $C_j$  is a disjunction (see Predicate logic for details on this and other forms to be described here). The second normal form is called Skolem normal form (SNF) which is PNF with the further requirement that all quantifiers be  $\forall$ . Any closed wff,  $F$ , can be transformed into another closed wff,  $\text{SNF}(F)$ , such that  $\text{SNF}(F)$  is in SNF and such that  $F$  is unsatisfiable iff  $\text{SNF}(F)$  is unsatisfiable. (Note:  $F$  and  $\text{SNF}(F)$  are not necessarily logically equivalent, i.e., they do not necessarily have the same truth value in every interpretation. But if one has an interpretation that makes it true, then the other does, and conversely. Also, the SNF is not unique.) Most theorem-proving programs work directly with the  $C_j$  formulas above. These are called literals. Each disjunct in a clause is either an atom or of the form NOT  $L$  where  $L$  is an atom. These disjuncts are called literals.

Two important notions for theorem-proving programs are soundness and completeness. Soundness means that any newly concluded formula must follow logically from the formu-

las used to deduce it. This means, in turn, that if a program derives a contradiction, the contradiction was not introduced by something the program did but had to have been a property of the original set of formulas that were given to the program. Completeness (qv) means that if the program was given a set of formulas that represented a theorem, the program could find a proof of it given enough resources. In the case of refutation procedures, if the program is given a set that is unsatisfiable (represents the denial of a theorem), the program will be able to derive a contradiction, given enough resources (refutation completeness). The key words are "given enough resources." As in other AI applications, a theorem-proving program may run out of time or memory while searching for a solution. Thus, it is questionable whether it is better to have a complete program or a program with good heuristics (qv) that may succeed more of the time.

The last notions from logic that play central roles in automated theorem proving are those of Herbrand universe and Herbrand's theorem. As noted above, to be a theorem, a formula must be valid in all interpretations. Herbrand (1) provided an important result to reduce consideration to a smaller class of interpretations. The terms and atoms built up only from the constant, function, and predicate symbols of the language being considered are variable-free and are called ground. The set of ground terms is called the Herbrand universe; the set of ground atoms is called the Herbrand base. A constant symbol can be added to the language if there is none to begin with, so the Herbrand universe is a nonempty set and can be used as the domain of an interpretation. The assignment of meaning to the constant, function, and predicate symbols can be done as follows: First, let each constant and function symbol represent itself so that the interpretation of each ground term viewed as a formula is just that term itself viewed as an object of the domain; next, pick a set of ground atoms out of the Herbrand base, and assign these atoms to be true so that a predicate symbol  $P$  is assigned the relation that is true for (ground) terms  $t_1, \dots, t_n$  just when  $P(t_1, \dots, t_n)$  is in the chosen set. Such an interpretation is called a Herbrand interpretation. Herbrand then proved the following result (recall that an SNF formula has the form  $\forall x_1, \dots, \forall x_n (C_1 \text{ AND } \dots \text{ AND } C_k)$  and that the main approach for programs is to show unsatisfiability).

**Herbrand's Theorem.** A set of clauses is unsatisfiable iff there are ground clauses  $C_1', \dots, C_m'$  such that each  $C_k'$  is obtained from a  $C_j$  by substituting ground terms for the variables and such that  $C_1' \text{ AND } \dots \text{ AND } C_m'$  is Boolean unsatisfiable.

Thus, in studying refutation procedures, one need concentrate only on Herbrand interpretations. Since the objects of such an interpretation are the terms of the language itself, such studies often reduce almost to the study of symbol manipulation, which in turn is directly related to the computer operations that will be applied to the formulas by the theorem-proving program.

## History

The earliest attempts at automated theorem proving were applied to propositional logic (qv) (2). The idea here was not so much to make programs that would be effective aids in mathematics or other applications; after all, propositional logic is decidable, but then also it is not very useful for expressing

mathematical or other concepts. Rather, the idea was to study heuristics in a symbol-manipulation program. This work soon led to other interesting developments in general problem solving (qv), but as a theorem-proving methodology it was quickly abandoned.

The first serious work aimed at first-order logic was begun around 1960 (3–5). These efforts were based directly on Herbrand's theorem. The programs would generate some finite initial set of the Herbrand universe, substitute these ground terms in all possible ways in the clauses, and attempt to show that the resulting ground set was Boolean unsatisfiable. If this was not possible, more ground terms were generated, etc. Much of this work was directed at fast Boolean CNF refutation procedures and at how to reuse work that had already been done in the next iteration. An example is the one-literal rule of Davis and Putnam (4). If one of the ground instance clauses,  $C_j'$ , consisted of only one literal, say  $L$ , any attempt to make the entire conjunction true would have to assign  $L$  true. In this case any clause containing  $L$  would also be true, and any clause containing the negation of  $L$  could only be made true by assigning one of its other literals true. The program could reduce the set of instances by deleting all those clauses containing  $L$  and by deleting the occurrences of NOT  $L$  from the rest. The reduced set is unsatisfiable iff the original set is. Of course, the difficulty with this approach is that the set of all ground instances is huge. For example, the associativity axioms in the formulation of group theory given above have six distinct variables in them. If there were 10 ground terms, then each one alone would have  $10^6$  ground instances.

The idea of working directly with the first-order clauses rather than with ground instances was developed independently by Robinson and Prawitz (6,7). The two divergent methods rest on the same concept—determine some potential manipulations for some first-order formulas; then determine if the necessary formulas can be made identical by an appropriate substitution. (Finding the right substitution is described below in the section on unification.) Consider the one-literal rule above and the general clauses  $P(x, e, x)$  and  $-P(r, s, u) - P(s, t, v) - P(u, t, w) P(r, v, w)$  representing right identity and associativity from group theory. If the variable  $x$  is substituted for both  $r$  and  $u$  and the term  $e$  for  $s$  in the second clause, the result is  $-P(x, e, x) - P(e, t, v) - P(x, t, w) P(x, v, w)$ . Note that this is an instance of the second clause, but not a ground instance. The clause  $-P(e, t, v) - P(x, t, w) P(x, v, w)$  can then be formed and in essence captures all of the possible one-literal rule applications to ground instances of the two clauses above in which the first literal in associativity is canceled by the one-literal clause. This alleviates the problem of generating all of the ground instances. Unfortunately, the above two clauses also admit cancellation using the second and third literals of associativity, so the original two clauses cannot be deleted after forming the new clause. Thus, rather than reducing the size of the clause sets as in the Davis–Putnam approach for ground sets, applying such rules actually increases the number of clauses present. However, this growth is much slower than the growth in the number of ground instances, and programs based on this approach can explore significantly more of the search space. This notion—performing a general inference on the first-order level to replace the corresponding set of inferences on the ground level—is perhaps the most important development for automated theorem proving. Of the two independent approaches above, the one by Robinson based on a rule of inference called resolution (6) became the focus of at-

tention from 1964 to around 1970. Resolution is described below and in the entry Resolution. In 1968 a similar rule, called paramodulation, was introduced for the special handling of equality (8).

Immediately following the introduction of resolution, there was a flurry of research (e.g., Refs. 9–14) on how to restrict its application because there are, in general, a great many more resolutions that can be made from a set of clauses than are used in a refutation, just as in most AI applications. A typical program might generate 2000 clauses, perhaps 10 or 15 of which contribute to the contradiction. More often, programs ran out of space before finding a refutation. During this period the overriding concern was to maintain completeness. (As noted above, this may have been less productive than emphasizing effectiveness.) The general idea was to impose some restrictions on the clauses used as parents or on the resolvents themselves to cut down on the large number of allowable inferences. Various simple strategies are described in the section on resolution.

During this same period there were a few groups (13,15,16) performing experimental studies to help determine effective strategies. It was not until much later (17,18) that serious comparisons between various proposals were made. The first programs were relatively straightforward implementations; the purpose was just to see how the rules and strategies worked rather than to develop production-quality systems. Typical problems run with the program were elementary algebra problems from group theory and others. Theorems such as “The square of every element equals the identity implies commutativity in a group” could be proved in under a second on the better implementations. Experimental effort concentrated on running programs with known theorems so that the program's progress (or lack of it) could be determined. During the sixties one bright spot was the discovery of a new result, SAM's Lemma (19). The SAM program (semi-automated mathematics) was given axioms for lattice theory and allowed to run for a long period of time. The users examined the new formulas that had been accepted by the program and found an interesting one. It would not be until considerably later that automated theorem provers would be used in a significant way to help develop new mathematics or other disciplines.

Several significant developments occurred around 1970. First, the entire field came under heavy criticism from the AI community at large because of the lack of progress and the dedication to completeness. It was suggested that logic itself might not be appropriate for a great many other AI problems such as cause-and-effect reasoning (see Reasoning, causal) and language understanding (see Natural-language, understanding). In any case the feeling was that resolution with complete strategies did not represent “intelligent” problem solving. Many of those in theorem proving began to reconsider the way resolution and other approaches were being used. This led to the development of different approaches to theorem proving (e.g., Refs. 20 and 21) as well as new strategies for resolution that emphasized the incorporation of user instinct into the program's control and allowed more flexibility in writing strategies (e.g., Refs. 22 and 23). A second major change was that several researchers began to pay close attention to implementation issues. Data-structuring techniques and new algorithms (e.g., Refs. 24 and 25) made previously costly techniques such as full-subsumption testing (see below) feasible and generally improved the performance of programs. This in turn allowed experiments on more difficult problems, which in

turn led to the development of more ideas for controlling the search, etc. This kind of activity led eventually to programs that have made useful contributions to mathematics and other application areas (21,26,27). A third important change occurring in the late sixties was the emphasis on problem domains other than mathematics. As early as 1968 (28,29), it was suggested that logic and theorem proving could be used for applications such as question answering (qv) and proving properties of programs. During the seventies a large number of applications were studied, some of which are discussed below in the section on applications. Even for those systems that were applied to mathematics, a major effort was devoted to special-purpose systems that could be made to perform well in a single area of mathematics (20) as opposed to being fully general-purpose. In all of these activities the idea was to find an application area and to take advantage of any special features or knowledge about that domain in the program, a more AI-like approach.

### Some Typical Theorem-Proving Methods

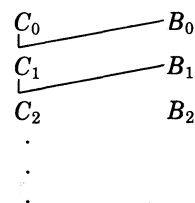
**Resolution.** The resolution (6) rule is a generalization of *modus ponens* and the one-literal rule. In its simplest, Boolean form it can be stated as follows: If  $C$  is the clause  $L \text{ OR } C'$  and  $D$  is the clause  $\neg L \text{ OR } D'$ , then deduce  $C' \text{ OR } D'$ . It is easy to see that this is a sound rule of inference by considering the possible truth-value assignments that make  $C$  and  $D$  both true. In the first-order case there are two additional factors. The first is that the program may have to find a unifying substitution in order to allow the deduction to go through. The second is that the program may have to pick sets of literals in the two clauses. As an example of this last feature, consider the two clauses  $P(x) \text{ OR } P(y)$  and  $\neg P(u) \text{ OR } \neg P(v) \text{ OR } R(u, v)$ . If one took instances in which all the variables were replaced by the same ground term, say the constant  $a$ , the resulting clauses would be  $P(a)$  and  $\neg P(a) \text{ OR } R(a, a)$ . Note that identical literals in a clause, e.g.,  $P(a) \text{ OR } P(a)$ , reduce to one literal. From the two instances, one can infer  $R(a, a)$ . To capture the family of such ground inferences, the program must find a unifying substitution for all the atoms  $P(x)$ ,  $P(y)$ ,  $P(u)$ , and  $P(v)$ . A similar rule for equality, called paramodulation (8), uses two clauses of the form  $s = t \text{ OR } C'$  and  $L(s') \text{ OR } D'$ , where  $s'$  indicates one occurrence of a target term in the second clause. The program attempts to match  $s$  with  $s'$  and, if successful, deduces the corresponding instance of  $C' \text{ OR } L(t) \text{ OR } D'$ . That is, the chosen target term is replaced by the term it is equal to. One can again verify that this rule is sound. It is also the case that resolution and paramodulation are refutation complete even with various restrictions to control their use. For a fuller discussion of resolution with examples, see Resolution. For a complete treatment, see Refs. 30–32.

Strategies for controlling the use of resolution and paramodulation can be generally classified in the following four-step scheme:

1. *Ordering the choice of clauses to attempt to resolve.* One example is choosing the two clauses that yield the smallest possible resolvent. In general, if the two parent clauses have  $n$  and  $m$  literals, respectively, the resolvent will have  $n + m - 2$  literals (except when some of the literals unify and coalesce as above). Because the program would be searching for a clause with no literals at all, it was felt that concentrating on small clauses would help more to direct the program. A variation is to emphasize unit clauses first (i.e., clauses with only one literal) using the smallest-first strategy as a secondary guide.

This is called the unit-preference strategy (9). The intuition here is that a unit clause is more like a fact, whereas a clause with several literals expresses a list of alternatives that must be true. Moreover, the resolvent is guaranteed to be shorter than the nonunit parent, again emphasizing the smallest-first aspect. The reader should note that these guidance strategies do not eliminate any valid resolvents from potentially being generated; they only reorder the way in which the space of all possible resolvents is explored.

2. *Restricting the pairs of clauses that are allowed to resolve.* One of the earliest and most important (and successful) of this kind of strategy is the set-of-support strategy (10). A typical set of clauses contains some clauses that are the axioms of the theory, some clauses that are the special hypotheses of the problem given to the program, and some clauses that represent the denial of the conclusion. If a program reasons from the axioms alone, it should be able to generate most, if not all, of the theory being studied, relatively little of which would be relevant to the given problem within that theory. The set-of-support strategy requires that each resolvent must involve in its history some clause from the denial or some clause from the special hypotheses. This prevents derivation of clauses from only the axioms resolving together directly. Notice that this does eliminate some otherwise legal resolvents from being generated. If the set of support is chosen so that its complement set is satisfiable, completeness is maintained. This will usually be the case when the set of support is the special hypotheses and denial of conclusion. A second restriction strategy developed in the late sixties is linear resolution (12). In this strategy the deduction as a whole is viewed as a line of inferences:



The restriction requires that each  $B_i$  clause come from the original set of clauses (i.e., be an input clause) or else be one of the preceding  $C_j$  clauses, i.e.,  $j < i$ . The idea was to simulate a common human strategy, namely, line-by-line reasoning, and to avoid deductions that look like highly branched trees with many independent lines of reasoning eventually merging. Of course, while searching for a linear refutation, the program might start several such chains of reasoning, but these chains are not allowed to interact. A number of other restriction strategies were proposed between 1965 and 1972, many of which proved to be of little use in producing programs that could effectively find refutations.

3. *Reformatting clauses after they are generated.* The major example of this kind of strategy is demodulation (33), later called rewriting. Consider the group-theory example again. The equality  $f(e, x) = x$  should hold for all values of  $x$ . Then a clause like  $P(a, f(e, b), c)$  should represent the same fact as  $P(a, b, c)$ . Using demodulation, a new clause is immediately simplified as much as possible by a special list of equalities. In this case a term in the new resolvent must be an instance of the complicated side of the demodulator as opposed to the two terms simply having a unifier (which could substitute in both clauses) as in paramodulation. Also, only the simplified version is saved (another distinction from paramodulation). Unlike paramodulation, however, there are situations where the



use of demodulation can block the program from finding a refutation for an unsatisfiable set. However, for most problems in mathematics as well as for problems in many other fields, the use of demodulation is indispensable. Thus, demodulation represents the first incomplete strategy to be used extensively in automated theorem proving. Part of the utility of demodulation is that it helps detect redundant information in the set of clauses, leading to the last kind of strategy.

4. *Deleting clauses.* Continuing the group-theory example above, suppose the program has in its memory the clause  $P(a, b, c)$  when the resolvent  $P(a, f(e, b), c)$  is generated. After demodulation, the new resolvent becomes identical to an existing clause. There is no advantage to retaining two identical clauses, so the program can throw the new resolvent away. In many cases the program will generate identical or less general clauses even without demodulation. For example, the program might have the axiom  $P(x, e, x)$  for right identity and generate clauses of the form  $P(a, e, a)$ ,  $P(f(a, b), e, f(a, b))$ , etc. In general, a clause  $C$  subsumes a clause  $D$  if there is some substitution  $s$  such that  $Cs$ , viewed as a set of literals, is contained in  $D$ . When this happens,  $D$  should be removed from the program's memory. When there are a large number of clauses, testing for subsumption between pairs can be expensive. It was not until the seventies (25) that efficient methods for doing this on a large scale were developed. However, even for simple cases, as when  $C$  and  $D$  are both unit clauses, the effectiveness of this strategy was recognized very early (15). In some mathematics problems as much as 90% of the clauses generated are redundant. In addition to subsumption, a second elimination strategy focuses on discarding any resolvent that contains both a literal and its negation. Such clauses, called tautologies, are always true and do not contribute toward the generation of falsehood, i.e., a contradiction.

**Unification.** One of the most basic operations in theorem proving, indeed in all AI, is to find a unifier for a set of formulas, i.e., a substitution for the variables in the formulas that makes them identical as strings. Moreover, a particular kind called a most general unifier (MGU) is usually required. An MGU does the least amount of substitution necessary to achieve the unification. For example, the two atoms  $P(a, x, y)$  and  $P(u, b, v)$  can be unified by the substitution  $s_1 = \{a/u, b/x, c/y, c/v\}$ , which maps both atoms into  $P(a, b, c)$ . However,  $s_2 = \{a/u, b/x, y/v\}$  also unifies these two atoms, with  $P(a, b, y)$  as the common instance. If a program applies  $s_2$  first, it can still generate  $P(a, b, c)$  (i.e., the result of applying  $s_1$ ) by applying a third substitution  $s_3 = \{c/y\}$ . However, the reverse is not the case; after applying  $s_1$ , there is no way to obtain  $P(a, b, y)$  by substituting for variables [indeed, there are no variables left in  $P(a, b, c)$ ]. This is crucial for certain problems. For example, consider the clauses

1.  $P(a, x, y)$ ,
2.  $\neg P(u, b, v) R(v)$ , and
3.  $\neg R(d)$ .

If an inference is made from clauses 1 and 2 on the  $P$  literals using  $s_1$  to match them, the result is  $R(c)$ , which does not produce a contradiction with 3. On the other hand, if  $s_2$  is used, the resolvent is  $R(y)$ , which leads to the empty clause.

A restatement of the original version of the unification algorithm from the first paper on resolution (6) is given below, followed by a discussion of some improvements. The algorithm

takes as input two atoms, although simple generalizations allow for arbitrarily many atoms to be unified at once. Let the atoms be  $A$  and  $B$ .

1. Let  $s_0$  be the empty substitution (i.e., the substitution that does nothing). Let  $i = 0$ .
2. If  $As_i$  and  $Bs_i$  are identical, stop with output  $s = s_i$ .
3. Otherwise, let  $t_1$  and  $t_2$  be the leftmost terms of  $As_i$  and  $Bs_i$  whose starting symbols are different.
4. If neither  $t_1$  nor  $t_2$  is a variable, HALT with FAILURE. If one is a variable that occurs in the other, HALT with FAILURE.
5. Otherwise, at least one of  $t_1$  and  $t_2$  is a variable. Let  $u$  be one that is a variable and let  $v$  be the other one of the two terms. (Note: they could both be variables, in which case the program can pick either one for  $u$ .) Let  $s_{i+1} = s_i * (v/u)$  be that substitution that produces the same result in one step as  $s_i$  followed by  $(v/u)$  for arbitrary terms (see Ref. 6 for the definition of  $*$ ). Increment  $i$ , and go to step 2.

Intuitively, the algorithm proceeds left to right and, at each place where the two terms are different, attempts to make them alike by replacing a variable by the corresponding term. The term is, in fact, the minimal term necessary to make the disagreeing position match. Of course, not every pair of terms can be made alike by substitution. Such nonunifiable cases are found in step 4. For example, the algorithm fails after two iterations, given the atoms  $P(a, x, x)$  and  $P(y, b, y)$ . In the first iteration  $y$  will be replaced by  $a$ , and in the second  $x$  will be replaced by  $b$ , after which the two atoms have been transformed into  $P(a, b, b)$  and  $P(a, b, a)$ . Similarly, no substitution can make the two atoms  $Q(x, x)$  and  $Q(y, f(y))$  alike. After one iteration step 4 will find the second failure case with  $u$  being the variable  $x$  (or  $y$  depending on how  $u$  and  $v$  were chosen at the first step) and  $v$  being  $f(x)$  [or  $f(y)$ ]. Robinson (6) also proved the following.

**Unification Theorem.**  $A$  and  $B$  are unifiable iff the unification algorithm stops at step 2, in which case  $s$  is a most general unifier.

Several improvements to the original algorithm have been made over the years. Among the most important is to eliminate the need to apply the substitution  $s_i$  at every step, thus saving considerable symbol manipulation. This improvement is accomplished by breaking the process into two separate steps—a left-to-right scan, called *match*, in which variables are tied to terms, and a second process, called *backsub*, in which the substitution is completed and the “occurs” test is performed. As an example, consider the pair of atoms  $P(a, x, g(y), x)$  and  $P(w, g(w), z, z)$ . In scanning left to right, note that  $w$  must become  $a$ , so the program “ties”  $w$  to  $a$ . Then  $x$  must be tied to  $g(w)$ . Note that the program does not substitute into the two atoms in this first stage, so the fact that  $w$  has already been tied to something has no immediate effect on  $x$ . Next  $z$  must become  $g(y)$ . At the last position  $x$  and  $z$  occur, but in preceding steps  $x$  and  $z$  have been tied to other terms, so that it is  $g(y)$  and  $g(w)$  that are compared, resulting in the mating of the two variables  $y$  and  $w$ . Of course,  $w$  has already been tied to  $a$ , so  $y$  becomes tied to  $a$  also. Note that this linkage can be maintained and used without actually substituting into the original two atoms. Any matings are recorded in a variable table, and the table is consulted when a variable is scanned. The result of this match scan is that  $x$  is tied to  $f(w)$ ,  $w$  and  $y$

are tied to  $a$ , and  $z$  is tied to  $g(y)$ . In the second phase the program actually looks inside the substitution terms and discovers, e.g., that the  $w$  in  $g(w)$  is actually  $a$  so that  $x$  should ultimately be replaced by  $g(a)$ . Situations in which a variable would have to be replaced by a term containing that variable are discovered in this stage. The vast majority of failures arise from nonvariable terms with distinct starting symbols, and these are discovered in stage 1.

**Connection Graphs.** One major and successful representation scheme is the connection graph (34,35). In this scheme nodes in a graph correspond to clauses with a subnode for each literal. Links are drawn between pairs of unifiable literals or terms to indicate potential deductions between the two containing clauses. The unifier of the two end terms or literals is attached to the link. This linkage is done once at the beginning and only updated as deduction proceeds. When a link is chosen for deduction, the corresponding clause is generated and possibly linked into the graph. The new links are inherited from the links attached to the parent clauses. The link that produced the deduction is deleted, preventing one kind of combinatorial redundancy. (There are other techniques for this for programs that do not use graphs and therefore do not delete links.) By deducing along certain kinds of paths, a graph prover can mimic the typical resolution strategies. However, more complex patterns yield very complicated strategies that would be difficult to code otherwise. For example, the MARK-GRAF program (36) has a variety of terminal patterns that it can look for. When linking in a new clause, the program might check if within, say, three levels of linkage all the literals of the new clause can be resolved away. Such a pattern is rather easy to notice in a graph representation, and since the unifiers are already attached to the links, determining whether the literals can be unified to complete the macro deduction is relatively easy and quick. For complicated patterns the corresponding technique for a nongraph program might not be easy to express or code into the system.

**Matrix Methods.** One of the early proposals that has received renewed interest is the matrix method (37,38). In these methods the formulas are arrayed in a matrix, with an implicit OR between elements along a row and an implicit AND between one row and the next. Some number of copies of each formula are placed into the matrix. A path through the matrix proceeds from top to bottom, picking out one formula in each row. If the formula is not a literal, it is represented as a submatrix, and any path through this submatrix is a path through the subformula. The program tries to determine if there is a substitution that will make every path through the matrix contain a complementary pair of formulas. The set of these complementary pairs is called a *mating*. As a simple example, consider the following set of clauses and matrix and mating:

$P(x)$	$P(x_1)$
$-P(y) \quad Q(y)$	$P(x_2)$ $-P(y)$
$-Q(a)$	$-P(z)$ $Q(y)$
$-P(z) \quad R(z)$	$R(z)$ $-Q(a)$
$-R(b)$	$-R(b)$

Note that two copies of  $P(x)$  with different variables for  $x$  have been included. Every path through this matrix has a pair of mated literals, and there is a substitution that simultaneously unifies all the pairs up to sign. Thus the above matrix (and therefore the set of clauses) is unsatisfiable. Note the difference from the Herbrand instantiation method. Here copies are made, not ground instances, and the substitution is found only after (or in some cases at the same time as) the mating is developed. If no unifiable mating is found, a matrix program has to seek new copies of formulas to expand the matrix, as indeed would have been the case if only one of  $P(x_1)$  and  $P(x_2)$  above had been used. The growth in the number of formulas is very slow compared to resolution. However, how to select what formulas to duplicate remains a serious problem for this method.

**Human-Oriented Systems.** Resolution and related rules are usually termed machine-oriented because the formulas must be put in a normal form not easily readable by humans and the rules themselves do not correspond to the way humans make deductions. Several theorem-proving systems have been developed that operate more like mathematicians. These systems typically have a relatively large number of inference rules, each representing a typical kind of human activity. For example, the system for proving mathematical theorems described in Ref. 20 is interactive and has operations such as "replace a defined symbol by its definition," "substitute a term for a variable," "explore a few more deductive steps," and "back up to a previous place in the proof search." This system also had an automatic mode in which the various rules were applied in a prescribed order. The formulas were input to the program in their full first-order form and were not necessarily Skolemized. The program typically worked on goals of the form  $H_1 \& \dots \& H_n \rightarrow C$ , where the  $H_i$  and  $C$  could themselves be complex formulas. When being used in the interactive mode, these systems were quite effective in aiding the user to find a proof, especially since the formulas displayed at the screen and the operations performed were indeed like those the user was familiar with. A difficulty in the automatic mode is how to select among and control the use of the many different rules.

**Complete Sets of Reductions.** An interesting and active area of research relating to equality theorem proving is complete sets of reductions (39). Equalities are oriented;  $s = t$  is written as  $s \rightarrow t$ , and the program is only allowed to replace instances of  $s$  by instances of  $t$ . The rationale is that many equalities represent reductions of complex terms to simple ones (a process originally called demodulation in theorem proving). It often happens that a set of reductions has two important properties:

**Confluence:** If  $t$  reduces to  $s_1$  and also to  $s_2$ , there is an  $s$  such that both  $s_1$  and  $s_2$  reduce to  $s$ . In effect, the order of reduction does not matter.

**Termination:** For no term can an infinite sequence of reductions be applied. This eliminates infinitely swapping arguments with reductions like  $f(x, y) \rightarrow f(y, x)$ .

If  $S$  has both the above properties, an equality  $t_1 = t_2$  is implied by  $S$  iff  $t_1$  and  $t_2$  reduce to the identical symbol string. This is a powerful tool, but of course one has to know that  $S$  is terminating and confluent. The Knuth-Bendix completion algorithm has been proposed for completing an initial set of reductions. Reductions are analyzed to see which pairs might lead to two

separate terms that are really equal in the theory. For example, using the two reductions  $f(a) \rightarrow b$  and  $a \rightarrow c$ , the term  $f(a)$  can reduce both to  $b$  and to  $f(c)$ . Of course, in a system where  $f(a) = b$  and  $a = c$ , and two reduced terms  $b$  and  $f(c)$  are also equal. Therefore, a new reduction, like  $f(c) \rightarrow b$ , should be included. If a set with no additional critical pairs, as terms like the above are called, is obtained, it will have the useful properties cited above. Unfortunately, some sets of reductions do not have complete extensions. Further, the algorithm may fail to complete a "completable" set in one of two possible ways: it keeps finding critical pairs, or some critical pair cannot be oriented. Fortunately, many theories have known complete sets of reductions and are therefore decidable. Of course, if one forms a subtheory by adding a new equality, the augmented set of reductions may no longer be complete (or even have a complete extension).

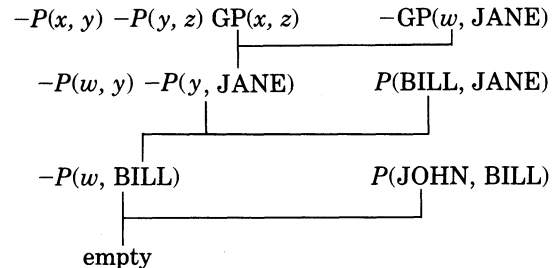
### Applications

In the various applications of automated theorem proving there are some common features. Most applications require careful thought as to how to represent problems, i.e., how should the user axiomatize the elements of the problem, what are objects of the domain that require names (constant symbols), what are the relationships (predicate symbols), etc. The user must also understand what problem feature is represented by a logical inference. For example, in mathematics inference corresponds quite naturally to mathematical deduction, while in circuit design a resolvent might represent a step in the construction of the final circuit. A refutation in a mathematics problem represents a proof that the positive statement of the conjecture is, in fact, a theorem, while refutations in a question-answering system are used more to extract answers to a question. An important comment related to interpreting the output of a theorem prover is that, as with other AI programs, a theorem prover often runs out of resources before finding a refutation. In the early days of theorem proving, this was looked on as failure. More recently, however, extensive use has been made of the output from runs that do not produce the empty clause. The list of clauses that were generated often provides insight into the problem even when a complete solution has not been found. In this regard, the program may be said to be operating like a colleague—sometimes the colleague can give you the answer to a problem you pose, sometimes the colleague can give you useful hints or ideas, and sometimes the colleague is of no use at all. Particularly in the mathematics applications below, where new results were discovered and proved, this assistant mode has proved extremely profitable.

The application of theorem provers to mathematics is an obvious first thought. As noted, in the early days of automated theorem proving, programs were used mostly on theorems whose proofs were already known; this allowed the progress of the program to be measured. In the seventies a major effort was undertaken by one group to use an automated theorem prover as a mathematical tool or colleague (as described above) for research in mathematics as opposed to theorem proving (26,40). The program was also used for purposes other than just finding a proof, e.g., building certain example algebras as counterexamples to conjectures or showing the structure of certain kinds of algebras (40). In such cases the theorem prover was used almost as an algebraic calculating machine.

In the late sixties it was noticed that additional information could be extracted from a proof besides just that the clause set

was unsatisfiable (28). This led to the use of theorem provers as question-answering programs. A simple example: Consider the clauses  $P(\text{JOHN}, \text{BILL})$ ,  $P(\text{BILL}, \text{JANE})$ , and  $\neg P(x, y) \rightarrow \neg P(y, z) \vee \text{GP}(x, z)$ , representing simple facts and rules about ancestry. Suppose one wanted to prove the theorem  $\exists w \text{GP}(w, \text{JANE})$ . In the spirit of resolution, the clause representing the negation of the conclusion is added, and a refutation is found:



Thus, Jane has a grandparent. But even more, resolution refutations are constructive in the sense that one can extract an example of the  $w$  that was shown to exist. In this case JOHN was matched with  $w$ , and so JOHN is a grandparent of JANE. If there were other clauses that led to other refutations, each one would lead to a grandparent of JANE. In these applications the clauses represent facts and definitions in a database (e.g., a list of ground-unit clauses giving the parent relation and some clauses defining views like grandparent or ancestor), and the theorem prover must find all proofs instead of just one so that all answers could be returned. Later, some techniques were proposed by which a theorem prover in one step captured a large number of resolutions over these ground-unit facts, thereby facilitating the finding of multiple proofs (41). By the end of the seventies the idea of storing the facts not as ground unit clauses but as a traditional relational database was being studied (42). Here the name of a database relation acts like a predicate symbol, and each tuple in the table represents one ground unit clause. The relations represent the conjunction of all these ground unit formulas, but in a different format from that of the clauses stored in the theorem prover's memory. The nonground unit clauses represent definitions of views, integrity constraints, etc. The role of the theorem prover is to obtain a formula containing only predicates stored in the database by undoing definitions involved in the query. This base clause can then be transferred to the database to be answered in the normal database way.

An application of interest to computer scientists is proving properties of programs, typically partial correctness (29). One represents the program in question as a flowchart consisting of various paths through computation and decision boxes. The effect on the variables of the program of each computation box can be axiomatized, e.g., by Hoare's method. Each branch of a decision can similarly be tagged with the predicate that causes the program to go down that branch. Input conditions on the variables are attached to the branch leading into the program and the output requirements to the exit branch. Each path has a point, called a cut point, to which the programmer attaches a formula, called an invariant, representing some condition that is supposed to always be true at that point in the program's execution. For example, a simple sort program might have an outer loop with  $I$  running from 1 to  $N - 1$ . The predicate for going around the loop is that  $I$  be less than  $N$ . The invariant attached to the path going around the loop might be that the elements from 1 to  $I - 1$  are the smallest in the array and are in order. A path theorem for a path from one invariant position

to a consecutive invariant position is a formula of the form  $ARITH \& INV \& C \& W \rightarrow INV'$ , where  $INV$  is the invariant at the beginning of the path,  $C$  is the conjunction of all conditions that must be true for the program to follow the path,  $W$  axiomatizes what happens when the program executes the path,  $INV'$  is the invariant at the end of the path, and  $ARITH$  is the set of axioms about the arithmetic functions and predicates used in the program (like  $<$ ,  $+$ , etc). If one could prove every path theorem, by transitivity of implication applied inductively, the input condition would imply the output condition; thus, if execution ever proceeded with valid inputs from the entry to the exit, the output variables would have the values specified by the output condition. The proving of the path theorems was to be done by an automated theorem prover. However, as the programs to be verified become large, the number and complexity of path theorems grows. Also, the style of theorems and proofs is very different for different kinds of programming languages. Those systems dealing with more precisely defined languages, like LISP, have met with considerably more success. The more successful systems use formulas other than clauses and employ specialized inference rules. Some have had remarkable success (21).

A related application is that of program generation (43). Here the symbols of the language include some number of predicates, functions, and constants that represent primitive operations in some programming language. There may, of course, be other symbols as well. The axioms represent both the effects of the primitive operations in the programming language and fundamental facts from the underlying theory of the application, e.g., facts about assignment statements, IF statements, etc., and axioms about integer arithmetic. To write a program that computes  $y$  with the property  $OC(x, y)$  (output condition) given  $x$  with property  $IC(x)$  (input condition), one has the theorem prover try to prove  $\forall x \exists y (IC(x) \rightarrow OC(x, y))$  from the axioms. As with question answering, a refutation is only the first step. One must then construct a program. A resolution step on a literal consisting only of programming language symbols becomes an if-then-else structure in the program with the predicate of the resolution as the test. Similarly, the construction through unification of complex terms over function and constant symbols of the language represents computations. Unfortunately, not every refutation yields a program because many of the resolutions will be over general axioms of the underlying domain theory, and these do not lead directly to program constructs. Some search strategies have been developed to find only acceptable refutations. As above, this application runs into severe difficulties when the required program is complex. Considerable research needs to be done before useful systems are available.

Another interesting application is to use theorem provers as a calculating device. Consider the following clauses:

FACT(0, 1)  
 -FACT( $x, y$ ) FACT( $s(x), *(s(x), y)$ )  
 -(FACT( $s(s(0))$ ),  $w$ )

along with suitable axioms for  $*$  (multiplication) and  $s$  (successor). Extracting the answer from a refutation is tantamount to finding three factorial. In logic programming one concentrates on the relationships among the entities of the program and is relieved of the tedious details of traditional programming (especially those concerning flow of control). In addition, al-

though less efficient, logic programs supposedly are easier to write and debug, so that a new program might be made operational in a much shorter time. It soon became clear that a complete lack of concern for the operation of the program could lead to totally infeasible programs (e.g., sorting arrays by testing random permutations for being in order). Many of these early misdirections have been dealt with, and logic programming is an active area of research as well as a useful tool in the AI and other communities. A more complete treatment of this important topic is given in Logic programming.

## BIBLIOGRAPHY

1. J. Herbrand, "Recherches sur la theorie de la demonstration," *Travaux de la Societe des Sciences et des Lettres de Varsovie, Classe III Sci. Math. Phys.* **33** (1930).
2. H. Gelerntner, J. Hansen, and D. Loveland, Empirical Explorations of the Geometry Theorem Machine, in E. Feigenbaum and J. Feldman (eds.), *Computers and Thought*, McGraw-Hill, New York, 1963.
3. P. Gilmore, "A proof method for quantification theory," *IBM J. Res. Devel.* **4**, 28-35 (1960).
4. M. Davis and H. Putnam, "A computing procedure for quantification theory," *JACM* **7**, 201-215 (1960).
5. H. Wang, "Towards mechanical mathematics," *IBM J. Res. Devel.* **4**, 224-268 (1960).
6. J. Robinson, "A machine-oriented logic based on the resolution principle," *JACM* **12**, 23-41 (1965).
7. D. Prawitz, H. Prawitz, and N. Voghera, "A mechanical proof procedure and its realization in an electronic computer," *JACM* **7**, 102-128 (1960).
8. G. Robinson and L. Wos, Paramodulation and Theorem Proving in First-Order Theories with Equality, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 4, American Elsevier, New York, pp. 135-150, 1969.
9. L. Wos, D. Carson, and G. Robinson, The Unit Preference Strategy in Theorem Proving, *Proceedings of the Fall Joint Computer Conference*, New York, pp. 615-621, 1964.
10. L. Wos, D. Carson, and G. Robinson, "Efficiency and completeness of the set-of-support strategy in theorem proving" *JACM* **12**, 536-541 (1965).
11. D. Loveland, "A simplified format for the model elimination theorem-proving procedure," *JACM* **16**, 349-363 (1969).
12. D. Loveland, A linear Format for Resolution, *Proceedings of the IRIA Symposium on Automatic Demonstration, 1968*, Versailles, France, Springer-Verlag, pp. 147-162, 1968.
13. D. Luckham, Refinement Theorems in Resolution Theory, AI Memo-81, AI Project, Stanford University, Stanford, CA, 1969.
14. J. Slagle, "Automatic theorem proving with renamable and semantic resolution," *JACM* **14**, 687-697 (1967).
15. L. Wos, G. Robinson, and D. Carson, Some Theorem Proving Strategies and Their Implementation, Technical Memo 72, Argonne National Laboratory, Argonne, IL, 1964.
16. J. Slagle and P. Bursky, "Experiments with a multipurpose, theorem proving heuristic program," *JACM* **15**, 85-99 (1968).
17. J. McCharen, R. Overbeek, and L. Wos, "Problems and experiments for and with automated theorem proving programs," *IEEE Trans. Comput.* **C-25**, 773-782 (1976).
18. J. Minker and G. Wilson, "Resolution refinements: A comparative study," *IEEE Trans. Comput.* **C-25**, 782-800 (1976).
19. J. Guard, F. Oglesby, J. Bennet, and L. Settle, "Semiautomated mathematics," *JACM* **16**, 49-62 (1969).
20. W. Bledsoe and P. Bruell, "A man-machine theorem proving system," *Artif. Intell.* **5**, 51-72 (1974).

21. R. Boyer and J. Moore, *A Computational Logic*, Academic Press, New York, 1979.
22. L. Henschen, L. Wos, and R. Overbeek, "A theorem proving language for experimentation," *CACM* **17**, 308–314 (1974).
23. J. McCharen, R. Overbeek, and L. Wos, "Complexity and related enhancements for automated theorem proving programs," *Comput. Math. Appl.* **2**, 1–16 (1976).
24. R. Boyer and J. Moore, Sharing of Structure in Theorem Proving Programs, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 7, Edinburgh University Press, Edinburgh, pp. 101–116, 1972.
25. R. Overbeek, "An implementation of hyper-resolution," *Comput. Math. Appl.* **1**, 201–214 (1975).
26. L. Wos, S. Winker, R. Veroff, B. Smith, and L. Henschen, "A new use of an automated reasoning assistant: Open questions in equivalential calculus and the study of infinite domains," *Artif. Intell.* **22**, 303–356 (1984).
27. W. Wojciechowski and A. Wojcik, "Automated design of multiple-valued logic circuits by automatic theorem proving techniques," *IEEE Trans. Comput.* **C-32**, 785–798 (1983).
28. C. Green, Theorem Proving by Resolution as a Basis for Question-Answering Systems, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 4, American Elsevier, New York, pp. 183–205, 1969.
29. Z. Manna, "The correctness of programs," *J. Comput. Syst. Sci.* **3**, 119–127 (1969).
30. C. Chang and R. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.
31. D. Loveland, *Automated Theorem Proving: A Logical Basis*, North-Holland, Amsterdam, 1978.
32. L. Wos, R. Overbeek, E. Lusk, and J. Boyle, *Automated Reasoning: Introduction and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1984.
33. L. Wos, G. Robinson, D. Carson, and L. Shalla, "The concept of demodulation in theorem proving," *JACM* **14**, 698–704 (1967).
34. S. Sickel, "A search technique for clause interconnectivity graphs," *IEEE Trans. Comput.* **C-25**, 823–834 (1976).
35. K. Blasius, N. Eisinger, J. Siekmann, G. Smolka, A. Harold, and C. Walther, The Markgraf Karl Refutation Procedure, *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, B.C., pp. 511–518, 1981.
36. G. Antoniou and H. Ohlbach, Terminators, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 916–919, 1983.
37. P. Andrews, "Refutations by mating," *IEEE Trans. Comput.* **C-25**, 801–806 (1976).
38. W. Bibel, "On matrices with connections," *JACM* **28**, 633–645 (1981).
39. G. Peterson and M. Stickel, "Complete sets of reductions for some equational theories," *JACM* **28**, 233–264 (1981).
40. S. Winker, "Generation and verification of finite models and counterexamples using an automated theorem prover answering two open questions," *JACM* **29**, 273–284 (1982).
41. D. Fishman and J. Minker, "PI-representation: A clause representation for parallel search," *Artif. Intell.* **6**, 103–128 (1975).
42. H. Gallaire and J. Minker (eds.), *Logic and Databases*, Plenum, New York, 1978.
43. R. Waldinger and R. Lee, PROW: A Step Towards Automatic Program Writing, *Proceedings of the First International Joint Conference on Artificial Intelligence*, Washington, DC, pp. 241–252, 1969.

L. HENSCHEN  
Northwestern University

**THEORETICAL FOUNDATIONS.** See Philosophical questions.

**THEORY FORMATION.** See Learning; Information retrieval.

**THERAPY PLANNING.** See Expert systems; Medical-advice systems.

**TOMOGRAPHY.** See Sensors.

**TOOLS AND SPECIAL-PURPOSE EFFECTORS.** See Manipulators.

**TOP-DOWN PROCESSING.** See Processing, bottom-up and top-down.

**TREE SEARCH.** See Problem solving.

**TRUTH MAINTENANCE.** See Belief revision.

## TURING MACHINES

Turing machines are named after the British mathematician, Alan Turing (1), who formulated their definition. They are not machines in the everyday sense but rather a mathematical construct useful in the theoretical study of the absolute limitations of computing devices (2–5). Essentially they represent a reduction of the logical structure of any computing device that could, in principle, be constructed.

To properly convey exactly what they are and what relevance they have to AI, it is useful first to carry out an analysis of algorithmic computation similar to the one carried out by Turing himself (1,6) in arriving at his definition.

### Analysis

Imagine a scenario in which a clerk carefully and accurately follows a finite, precise, deterministic recipe for performing an arbitrary, discrete, symbol-manipulation task on some sheets of paper. The rules one learns with conscious effort in childhood for doing sums and products could be put in the form of such a recipe as, presumably, could the rules one learns without conscious effort for converting sentences from active to passive voice. The symbols manipulated could, e.g., be numerical, textual, or the "discretized" components of a picture (pixels). Assume the recipe leaves no room for alternative interpretations. It will even have to specify exactly where to look and write on the sheets of paper. In addition, assume the sheets are carefully ruled into little squares, each capable of containing a blank space or a symbol. If the clerk had access to a discrete data space topologically more complicated than sheets of paper, the graph-theoretic (7) structure of that space would be representable (without overlaps) in ordinary three-dimensional Euclidean space, and, hence, on the discretized, essentially three-dimensional structure of the stack of sheets

of paper with each sheet ruled into little squares—successive pages represent, then, successive two-dimensional, discretized planes. Hence, the scenario represents the most general case of algorithmic symbol manipulation.

In the next phase of the analysis certain seemingly severe restrictions are made on the forms of both the recipe and the sheets of paper. However, these restrictions do not limit the class of symbol-manipulation tasks that can be performed.

In general, the recipe could contain complex instructions that require the clerk to “parallel process,” i.e., to attend to and perhaps modify in one step a large number of symbols in little squares in many places on the sheets of paper. Any such complex instruction can be replaced by, perhaps, a long and tedious sequence of simpler instructions, each of which merely requires the clerk to attend to and perhaps modify (or change) the symbol in a single square. This sequence of simple instructions achieves the same symbol-manipulation effect as the complex instruction. Instructions in the recipe are restricted to this simpler type and each complex instruction is assumed to have been replaced by an appropriate sequence of these simpler instructions.

Imagine a two-dimensional configuration of symbols on one of the sheets. These symbols may be thought as forming the pixels of a scene. Imagine further a suitably wide second sheet of paper that contains, along one of its rows, the rows of the “scene” from the first sheet, laid end-to-end, with a special marker symbol between each row. Any recipe instruction for manipulating the scene on the first sheet can be converted to a suitable sequence of simple instructions for producing the analogous manipulation on the one-dimensional representation of the scene on the second sheet (the details of such a conversion are somewhat tedious, however). Similarly, but more generally, any recipe for performing symbol-manipulation tasks on the sheets of paper can be converted to a longer recipe for performing the analogous task on a one-dimensional paper tape (if more tape is needed in the course of a computation, it is added to the ends). The sheet(s) of paper are modified to consist of a single, one-dimensional, extensible tape. Assume the recipe has been appropriately converted. Some of the details of converting, e.g., from a two-dimensional to a one-dimensional tape are given in Ref. 8.

The recipe may still contain instructions for traveling a large number of tape squares (from the one the clerk is looking at) to the right or left along the tape. Consider an instruction that says to travel 42 squares to the right. It can be replaced by a sequence of 42 simpler instructions each of which says to move one square to the right. Similarly, any move instruction can be replaced by a sequence of instructions each of which are instructions to move one square right (or left). Move instructions in the recipe are restricted to be of this simpler type. Assume the recipe has had any other move instructions replaced by appropriate sequences of simple move instructions.

Since the recipe is finite, it can contain only finitely many instructions. Since the recipe is precise and deterministic, it must clearly specify the instruction with which to begin. One instruction, then, is designated as the start instruction. For the same reasons, the recipe must clearly specify which instructions, under which circumstances, are to follow which other instructions. The only circumstances that need be taken into account in deciding what instruction to do next are the current instruction and the (possibly blank) symbol currently scanned: a finite sequence of past symbols scanned can be “remembered” because this sequence can, in effect, direct the

clerk to pass to an instruction that is particular to it. The recipe instructions are restricted, then, without loss of generality, each to specify the next instruction only on the basis of the symbol currently scanned.

The only other effect an instruction need achieve is the modification of the symbol currently scanned. The choice of the modification specified in an instruction is limited, again without loss of generality, to depend only on which symbol is currently scanned.

There are finitely many instructions in the recipe, one of which is the start instruction, and each of which specifies (a) a move one square right or left and/or a symbol modification and (b) a next instruction, where both (a) and (b) are a function of the symbol scanned. An instruction will, of course, specify the exact dependence on the symbol scanned. Since the recipe is finite, it can mention only a finite number of different symbols. Therefore, each instruction can involve only a finite number of symbols. Hence, it can be assumed that each instruction indicates the action to be taken and the next instruction to be carried out for each of a finite number of possible symbols scanned. Note, e.g., in the process of converting the recipe to contain only these types of instructions, an unrestricted instruction, saying to pick up a symbol *S* in one place and put it in another, would be replaced by a sequence of restricted instructions that involves changing an *S* to a blank, followed by a series of moves, followed by changing a symbol to *S*.

The conclusion of this second phase of the analysis is that recipes, operating on a one-dimensional tape and containing only the restricted type of instruction of the previous paragraph, are just as general purpose at performing symbol-manipulation tasks as the unrestricted recipes operating on multiple sheets of paper with which one began: They can handle the same class of symbol-manipulation tasks although they may require more instructions to do it.

### Definition of Turing Machines

Imagine now that each instruction in the recipe (containing the restricted type of instruction) is replaced by a uniquely corresponding, internal state of a machine. The machine is to have no states not corresponding to an instruction. Suppose this machine is connected by a read-write head to an extensible, one-dimensional tape. Suppose the machine is wired so that it begins operation in the internal state that corresponds to the start instruction. Suppose further that when, at a given time, the machine is in any one of the finitely many states corresponding to a recipe instruction, it first carries out the tape operation (moving and/or changing a symbol) specified by that recipe instruction and secondly goes into the internal state corresponding to the next recipe instruction. The choice of tape operation carried out and the next state/instruction, of course, depends on the symbol the machine is scanning just previously, and the particular dependence was specified in the recipe instruction itself. Clearly such a machine directly simulates the action of the clerk operating with the modified recipe on the one-dimensional tape. Hence, this machine performs essentially the same symbol-manipulation task as the original unrestricted recipe operating on the sheets of paper. One would expect, since the original algorithmic symbol-manipulation task was arbitrarily chosen, that such machines are capable of performing any algorithmic symbol-manipulation task.

The Turing machines are (by definition) the machines obtained in this way. They could actually be built out of a kind of



switching network, called a McCulloch–Pitts nerve net (9). One can consider Turing-machine states also as analogous to states of mind (see Philosophical questions). Careful, mathematical definitions and examples can be found in Refs. 2–4.

### Applications

Turing's analysis, similar to the analysis above, was the crucial evidence to legitimize the definition of the algorithmic symbol-manipulation tasks as exactly those symbol-manipulation tasks performable by Turing machines (see Church's thesis). Having such a definition enables one to rigorously, mathematically prove that certain tasks cannot be done by algorithm. The technique is to show they cannot be done by any Turing machine. The point of making all those restrictions on recipes and paper was to arrive at a simple kind of machine that would be easy to work with mathematically. The point was not to devise a kind of machine that would be useful to build; the restrictions on Turing machines make them too awkward to be practical devices for actually running computations.

An example task (about Turing machines themselves) is described that does not have a chance of algorithmic solution. Clearly any given Turing machine started on an initially blank tape either eventually prints a given symbol *S* or it does not. Consider the problem of designing an algorithm to decide from a description of any Turing machine *M* whether *M* eventually prints *S* if it is started on an initially blank tape. Turing showed that no Turing machine can correctly make this decision about arbitrary *M*; hence, no algorithm can be found to do it.

Turing also indicated how to construct universal Turing machines. A universal machine, given a description on its tape of any Turing machine, simulates that machine. It is generally believed that von Neumann's (10) idea for stored-program computers was inspired by Turing's universal machine. Stored-program computers keep one from having to rewire a computer each time a new (symbol-manipulation) task is to be performed. The computer is merely asked, in effect, to simulate a machine for the task; the "machine" to be simulated is typically described by a program.

### Cognitive Science, Mechanism, Learning, and AI

Suppose that the highest level brain processes to which human conscious and unconscious thoughts/symbol manipulations are reducible are algorithmic and that these brain processes are really produced by precise, finite, deterministic recipes somehow "wet-wired" into the human brain. Then human cognition is simulatable by Turing machines. Hence, any limitative results about Turing-machine computations apply to humans too, and perfect computer modeling of human cognition is, in principle, possible. This form of mechanism is a principal assumption of modern cognitive science (qv) and implies that artificial intelligence can, in principle, do anything natural intelligence can.

It can be argued at least at the atomic level, that brain processes are random, not deterministic. Of course, they still might be deterministic at higher levels. But suppose they are not. What, then, can be salvaged of (this form of) mechanism? It can be shown (11) that if the probability distribution function of a random process is algorithmically computable, its expected or most probable behavior is, again, Turing-machine

simulatable. Hence, mechanism and all its consequences apply, in this case, to expected behavior.

What about human creativity (qv)? How does mechanism account for the unbidden images that occur to people and lead to solutions of difficult problems and/or works of great beauty and significance for the human condition? The mechanist would argue that humans are not consciously aware of the brain processes that invoke such insights; hence, one has the illusion that they are not algorithmically produced. One's conscious thoughts are the mere tip of an iceberg; the curious thing is that one has any thoughts other than unconscious.

When one learns a list of data, to ride a bicycle, etc., or even if one has had an interesting thought, one is presumably changed by the experience. A learning machine is just a machine that changes itself, perhaps in part on the basis of externally obtained information. If the changes are of bounded size, ordinary Turing machines will serve as a mathematical model—the changes are mere changes of machine state. A more interesting case allows changes of unbounded size. (Learning is thought of as a growth experience.) This can be mathematically modeled by two-tape Turing machines where one of the tapes is inside the machine. Mathematically it really makes no difference where the boundaries between inside and outside are drawn, and most of the recent theoretical literature on absolute limitations of learning machines (12,13) (see also Inductive inference) does not bother to draw such boundaries; in fact, much of it proceeds on a level somewhat more abstract than the Turing machine model. So far, there has been very little interaction between this theoretical literature and the recent literature in AI on implementing learning machines (14) (see Learning machines).

### Feasible Computations and AI

Many theoretical computer scientists investigate the inherent difficulty of useful computation tasks (15). These days the standard model of feasible computation is based on resource-bounded, multitape Turing machines, Turing machines that operate a finite number of one-dimensional tapes and are required to finish before using some preset amount of time and/or paper. For example, a task that is, in principle, algorithmically performable, nonetheless, will be infeasible if any multitape Turing machine for the task, performing one operation every nanosecond, generally requires longer than the expected lifespan of the sun to complete. Unfortunately, there are results from theoretical computer science (15) (see also Limits of artificial intelligence; Search) to the effect that many of the tasks of importance to AI are inherently infeasible. This presumably requires AI workers to look for partial or approximately correct algorithmic solutions to such tasks (see Heuristics).

### BIBLIOGRAPHY

1. A. M. Turing, "On Computable numbers with an application to the Entscheidungs problem," *Proceedings of the London Mathematical Society*, **42**, (1936), 230–265.
2. M. Davis and E. Weyuker, *Computability, Complexity and Languages*, Academic Press, New York, 1983.
3. H. Lewis and C. Papadimitriou, *Elements of the Theory of Computation*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

4. H. Rogers, *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, New York, 1967.
5. A. M. Turing, "Computing machinery and intelligence," *Mind* 59, 433-460 (1950).
6. M. Minsky, *Computation: Finite and Infinite Machines*, Prentice-Hall, Englewood Cliffs, NJ, 1967.
7. F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA, 1969.
8. J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.
9. M. Arbib, *Brains, Machines and Mathematics*, McGraw-Hill, New York, 1964.
10. J. von Neumann, *Theory of Self-Reproducing Automata*, edited and completed by A. Burks, University of Illinois Press, Urbana, IL, 1966.
11. E. de Leeuw, C. Moore, C. Shannon, and N. Shapiro, Computability by Probabilistic Machines, *Automata Studies, Annals of Math. Studies*, 34, Princeton University Press, Princeton, NJ, 1956.
12. J. Case, Learning Machines, in W. Demopoulos & A. Marras (eds.), *Language Learning and Concept Acquisition*, Ablex, Norwood, NJ, 1986.
13. D. Osherson, M. Stob, and S. Weinstein, *Systems That Learn: An Introduction for Cognitive and Computer Scientists*, MIT Press, Cambridge, MA, 1986.
14. R. Michalski, J. Carbonell, and T. Mitchell, *Machine Learning: An Artificial Intelligence Approach*, Tioga, Palo Alto, CA, 1983.
15. M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, CA, 1979.

J. CASE  
SUNY at Buffalo

## TURING TEST

### Can Machines Think?

In 1637 Descartes argued that a machine can never think (1). Descartes believed that even if a machine resembled a human, there are two certain tests by which a machine can be distinguished from a rational human being. First, although a machine may utter words, a machine can never reply appropriately to everything said in its presence in the way that a human can. Second, although a machine may perform certain things as well or even better than a human, a machine cannot have the diversity of action that a human has. Of course, many supporters of AI would challenge the inevitability of these results (see Limits of AI; Social issues of AI). But how would one demonstrate that a machine can think? What would have to be done in AI, at least in principle, in order to show that a machine can think?

The question "Can machines think?" is difficult to answer because its wording is vague and ambiguous. What do the words "machines" and "think" mean? If humans are regarded as meat machines, then obviously machines think. On the other hand, if being a machine and thinking are incompatible concepts, thinking machines are conceptually impossible. On this interpretation, computers may think, but they would cease to be machines if they did. Even the seemingly innocuous word "can" suggests an ambiguity about the question. Is the question "Are machines able to think today?" or "Will machines eventually have the ability to think?" or "Regard-

less of what happens technologically, is it empirically possible that machines think?"

### The Imitation Game

Turing (2) realized that the question "Can machines think?" is ambiguous and sought to replace it by less ambiguous questions about a game he called "the imitation game." He introduced the imitation game by imagining a version that is played by three human beings—a man (A), a woman (B), and an interrogator (C). The interrogator, who is in a room apart from the other two, asks each of them questions and tries to determine from their typewritten answers which is the man and which is the woman. The object of the game for the man is to imitate a woman, and the object for the woman is to inform the interrogator about her true sex. Hence, the man tries to deceive the interrogator by giving answers that a woman would give; whereas the woman attempts to help the interrogator by giving answers that would identify her as the woman.

Turing proposed a variation of this imitation game, now known as "the Turing test," in which a computer takes the part of A in the game. An interrogator in a separate room asks questions of A and B and tries to determine from their answers which respondent is the computer. In this version of the game the computer gives deceptive answers. When asked to add 34,957 to 70,764, Turing said the computer pauses about 30 s and then gives the answer 105,621. Rather than give a correct answer quickly, the computer gives an incorrect answer slowly to imitate what a human thinker might do. Unfortunately, Turing did not carefully describe the role of B in his new version of the game. Is the computer imitating B *qua* woman, *qua* human, or *qua* thinker? Moreover, other details of the game are not given. Does one respondent know what the other respondent has said? When is the game over? How many questions can be asked? Is the winner a respondent or the interrogator? Perhaps, the programmer wins! In any event, these niggling details are probably unimportant to Turing's main claim that questions about how well a computer does in the imitation game should replace the original question "Can machines think?"

Turing's goal was to make the issue of machine thinking more objective. As Turing pointed out, it would be absurd to take an opinion poll to determine whether machines think. Turing wanted a more scientific approach. The imitation game is Turing's scientific test, and it has the strengths of a scientific test. The Turing test provides a format for impartially comparing the behavior of humans with the behavior of computers. The results of the Turing test are repeatable and objective. Moreover, the Turing test emphasizes evaluation of intellectual ability and eliminates prejudice based on appearance. Turing did not regard his test as a necessary condition for attributing thinking to a machine, for a machine might think but do so too quickly and accurately to play the game well. But Turing did regard the test a sufficient condition for attributing thinking to a machine. Turing, unlike Descartes, thought that machines will become quite sophisticated in their linguistic abilities. Turing (3) asserted that in about fifty years' time it will be possible to program computers, with a storage capacity of about  $10^9$ , to make them play the imitation game so well that an average interrogator will not have more than 70% chance of making the right identification after five minutes of questioning.

Deep philosophical issues are not easily dismissed. When the philosophical question about machines thinking is converted into a scientific test like the Turing test, the underlying philosophical concern does not vanish but merely reappears in a new way. In this case the philosophical issue becomes "Is the Turing test a reasonable replacement for the question about machine thinking?" Turing was aware of this difficulty. On the one hand, Turing's inclination was to dismiss the question "Can machines think?" because he thought the question was "too meaningless to deserve discussion." On the other hand, he knew that questions about the adequacy of his replacement lurked in the background. Turing said, "We cannot altogether abandon the original form of the problem, for opinions will differ as to the appropriateness of the substitution and we must at least listen to what has to be said in this connexion" (3). Indeed, a considerable amount has been said in this regard since Turing proposed his test.

### Nature of the Turing Test

Discussions of the Turing test often assume that the test is based on some form of behaviorism or operationalism. Millar states that a virtue of the Turing test is that "it constitutes an operational definition which, given a computer terminal system, can be used as a criterion" (4); Searle dismisses the Turing test on the grounds that it is "unashamedly behavioristic and operationalistic" (5). Operationalism and behaviorism are views that claim that mental terminology, such as "thinking," should be defined in terms of overt behavior or dispositions to behave. The advantage of giving operational definitions of mental processes in terms of behavior is that overt behavior provides a deductive basis for claiming the existence of mental phenomena. If thinking is explicitly definable in terms of certain behavior and computers exhibit this behavior, then by definition computers think. But clearly a behavioral analysis of a mental concept like thinking is inadequate. Thinking is an internal activity, and fortunately for the civility of human interaction, human thinking does not have an immediate or certain manifestation in behavior or vice versa. A behavioral analysis of thinking may offer an easy justification of the Turing test, but it also makes it vulnerable to easy refutation.

Gunderson offers this kind of refutation in the form of a parody (6). He notes that the question "Can rocks imitate?" is perhaps "too meaningless to deserve discussion." So he imagines a toe-stepping game. In this game a person in one room puts his foot through a hole in the wall near the floor. In the next room there is a human and also a rock box apparatus that consists of a box filled with rocks coupled to an electric eye and releasing mechanism. The person with his foot through the hole in the wall must decide whether a human or rock box apparatus is pressing on his toe. Gunderson thinks his parody "lays bare the reason why there is no contradiction involved in saying, 'Yes, a machine can play the imitation game, but it can't think.' It is for the same reason that there is no contradiction in saying, 'Of course a rock box of such-and-such a sort can be set up, but rocks surely can't imitate'" (7).

Gunderson's expression of his parody is misleading in that it is not the rocks alone which imitate but the rock box apparatus. Moreover, a good behaviorist would point out that imitating is a complex activity which involves different kinds of behavior beyond that described in the toe-stepping game. Nevertheless, Gunderson's basic point that neither imitation nor

thinking can be completely captured by pointing to net results is correct. Thinking, after all, is primarily an internal activity.

**An Inductive Interpretation.** However, interpreting the Turing test in the light of operationalism or behaviorism is unnecessary. Turing himself does not say he is giving an operational definition, and he does not argue for behaviorism. Because the operational/behavioristic interpretation of the Turing test makes it so vulnerable to criticism and is not explicitly defended by Turing himself, it is reasonable to look for a more interesting interpretation of the test. Another interpretation of the Turing test is to regard it as an inductive test (8). On the inductive interpretation the Turing test provides a format for gathering inductive evidence that computers think. Just as in physics a subatomic target is bombarded with accelerated particles to reveal its nature; in the Turing test a target mind is probed with questions in order to reveal its nature (9). Inductive evidence gathered from accelerator tests or Turing tests might be weak or strong, but such evidence would never be deductively certain. Inductive evidence at most provides good reasons for believing that a particular hypothesis is true or false. As with any interesting scientific hypothesis there is no logical contradiction in saying that the evidence for it is true but the hypothesis is false. Under the inductive interpretation Gunderson's previous objection does not apply. Inductively speaking there is no contradiction in saying that a machine can pass the Turing test but it cannot think.

An inductive interpretation of the Turing test is aligned with a common sense as well as a scientific approach to knowledge. Ordinary knowledge that other humans think and how they think is generated by inductive inferences (qv) based on their behavior. Somebody is judged to think deeply or not so deeply about chess on the basis of that person's chess playing. On a common sense level nobody confuses thinking about a chess move with an actual move. The latter is evidence for the former. In numerous other kinds of human activities human behavior is regarded as inductive evidence for inner mental activity. An inductive approach is a natural way to investigate other human minds and seems like an equally appropriate way to investigate computer minds.

In summary, the inductive interpretation of the Turing test avoids the pitfalls of behaviorism and accords well with our scientific and common sense understanding of gathering knowledge. In addition, this interpretation provides the right framework for considering the contemporary debate about the adequacy of the test. A wide variety of objections to the Turing test have been raised over the years, and Turing himself discussed some of these objections which are based on everything from theology to ESP (2). But the central objections to the test are most interesting and forceful if they are understood as challenges to the inductive strength of the Turing test. Is the Turing test too easy, too narrow, or too shallow to establish that digital computers think? Here each of these objections is considered in the form of a criticism and series of replies.

### Is the Test Too Easy?

**Criticism.** The Turing test is not a severe test. Verbal skills are rather easy to mimic with a computer, and people are easily deceived by such superficial mimicry. Therefore, the evidence gathered in the Turing test would be insufficient to justify the conclusion that computers think.

Block maintains that humans "may be too easily fooled by

mindless machines" (10). Block cites Weizenbaum's program ELIZA as an example. ELIZA contains both a language analyzer and a script, which allows it to improvise a conversation around a certain theme such as cooking eggs or managing a checking account. With a script for Rogerian psychotherapy, the program is called "DOCTOR." Weizenbaum (11) reports, "I was startled to see how quickly and how very deeply people conversing with DOCTOR became emotionally involved with the computer and how unequivocally they anthropomorphized it" (12). Weizenbaum says that even his secretary, who had watched him work on the program, wanted him to leave the room while she conversed with the computer. "What I had not realized," says Weizenbaum, "is that extremely short exposures to a relatively simple computer program could induce powerful delusional thinking in quite normal people" (13). Although Block does not believe ELIZA would fool an inquisitive interrogator very long, he does think that "human gullibility being what it is, some more complex (but nonetheless unintelligent) program may be able to fool most any human judge" (14). Block argues that a computer might pass the Turing test if its program looked up replies from a large, but finite, set of stored conversations.

**Replies.** First, sophisticated language ability is not easy to model on a computer. ELIZA has a very limited ability to understand language. Although many improvements have been made in natural-language processing since ELIZA was developed, no existing computer possesses anything like the linguistic ability of a typical human being. Turing was right to see language skill as a measure of thinking, for it is sophistication in language that clearly separates humans from other animals and, at least so far, from computers.

Second, the knowledge required by a computer to pass the Turing test is enormous. In order to pass the test, a computer really needs the knowledge of a typical human. In the Turing test a computer must converse sensibly about a wide range of subjects from poetry to the weather. No existing computer system begins to have a knowledge base that is sufficient to pass the Turing test. Storing potential conversations in a computer is a logically possible, but highly improbable, method of producing a computer system that would pass the Turing test in real time.

Third, a serious interrogator would not be fooled easily. The interrogator's objective is to identify the computer and to falsify the hypothesis that a computer can imitate a human thinker. Therefore, the interrogator would ask a variety of difficult questions and would not engage in idle anthropomorphizing. Of course, an interrogator might be fooled, but this is the nature of inductive testing.

### Is the Test Too Narrow?

**Criticism.** The Turing test may be difficult to pass but still it is a test of only one activity—playing the imitation game. Surely, if computers think, they must be able to do more than play the imitation game. Tests beyond the Turing test must be administered in order to gather adequate inductive evidence that computers think.

Gunderson (15) compares the situation to a vacuum-cleaner salesman who claims that his vacuum cleaner is all-purpose but only demonstrates that the vacuum cleaner picks up bits of dust. To show that the vacuum cleaner is all-purpose, other

kinds of abilities must be demonstrated. Similarly, the computer must do more than play one game well. As Fodor (16) puts the point, "Turing would presumably have been dissatisfied with a device that could answer questions about how to boil water if it routinely put the kettle in the icebox when told to brew the tea."

**Replies.** First, characterizing the Turing test as just one test is a misleading numbers game. The Turing test is really a format for conducting many tests. By asking questions, an interrogator can test a wide range of linguistic abilities from making jokes to using foreign languages. Moreover, through language any subject matter can be discussed. An interrogator can test for thinking about urban renewal, sailing, loving, or the merits of playing Mah-Jongg in the late afternoon.

Second, a judgment about any scientific hypothesis based on a body of evidence may be overturned when additional evidence is found. Science is fallible by nature. But this does not mean that additional evidence must always be gathered before a justified inductive inference (qv) can be made. Otherwise, scientists could never gather enough evidence for any hypothesis. Thus, although evidence gathered outside the Turing test might alter a justified inductive inference that a computer can think, it does not follow from this that additional evidence is necessary in order to make a justified inductive inference that a computer thinks. A vacuum cleaner might be justifiably judged to be all-purpose if it picked up bits of dust and came with an assortment of attachments for doing other kinds of cleaning. A buyer would not have to watch the vacuum cleaner with a cobweb attachment actually suck up cobwebs before being able to reasonably infer it could do it. Similarly, one might reasonably infer from linguistic evidence that a computer played poker well without ever seeing the computer play a poker hand with traditional cards. Linguistic evidence, though fallible, is sufficient for a good inductive inference that another being thinks. For example, intelligent communication with unseen aliens on a distant planet would be a sufficient inductive basis for attributing thinking to them. Of course, such aliens might really be machines.

Third, even if further tests yielded some contrary evidence, the computer might be regarded as a thinker. Suppose that a computer that passes the Turing test routinely puts the kettle in the icebox when told to brew the tea. Perhaps, the computer thinks iceboxes are funny-shaped stoves, or perhaps its motor mechanisms are malfunctioning, or perhaps the computer is playing a joke. The hypothesis that a computer thinks is compatible with some evidence that appears to refute it. Of course, if the computer acted inappropriately in lots of ways, one might indeed reject the claim that the computer thinks. But rejection is the possible fate of any scientific hypothesis.

### Is the Test Too Shallow?

**Criticism.** The Turing test is too shallow because it provides behavioral evidence but no evidence about the internal mechanisms that produce this behavior. One's inductive inferences that other humans think are based not only on the fact that they behave similarly but on the fact that they are made similarly. Electronic computers are simply made of the wrong stuff to think. Computers might simulate but they cannot duplicate thinking. As soon as it is known that behavior is produced by a computer, one's explanation will shift to a physical account of

how the behavior comes about (17). Thus, the Turing test is inadequate because it hides information that would lead one to reject the claim that a computer thinks.

Searle (18) gives a version of this criticism with his Chinese-room argument. Searle offers a thought experiment in which he, who knows no Chinese, is locked in a room with a large batch of Chinese writing (a script). He is given a second batch of Chinese characters (a story) along with instructions in English that allow him to correlate the second batch with the first. Finally, he is given a third batch of Chinese characters (questions) along with more instructions in English that allow him to correlate the third batch of symbols with the first two. Searle imagines that he can identify and manipulate these symbols in a completely formal way (by shapes alone) with such care that from the point of view of somebody outside the room his answers to the questions are indistinguishable from those of a native Chinese speaker. Of course, he can also answer questions in English since he is a native English speaker. The difference, Searle claims, is that when a question is asked in English, he understands the question and his answer, but when the question is in Chinese, he understands neither the question nor the answer.

Searle thinks his example shows that there could be two systems that pass the Turing test but only one of which understands. Searle concludes that "it is no argument against this point to say that since they both pass the Turing test they must both understand, since this claim fails to meet the argument that the system in me that understands English has a great deal more than the system that merely processes Chinese" (19). Searle believes that any digital electronic computer will similarly lack understanding if it operates only in a purely formal way. Searle's criticism is not about the narrowness of the Turing test. Even if the computer were inside a robot that performed many activities, such as perceiving, walking, hammering nails, and eating, the robot would still be doing it all on the basis of manipulation of formal symbols and, therefore, according to Searle, would lack understanding. Searle believes that human beings understand because of the causal powers of the human brain. Computers do not understand because "syntax alone is not sufficient for semantics, and digital computers insofar as they are computers have, by definition, a syntax alone" (20).

**Replies.** First, although humans are biologically alike, it is simply not the case that one's inferences that others think are based on one's knowledge of their internal operation. Not even brain scientists examine the brains of their friends before attributing thinking to them. Obviously, brains are crucial for thinking; but it is easy to exaggerate one's dependence on information about our biological workings when attributing thought to others. Aristotle knew that others thought, but he also believed that the brain's function was to cool blood.

Second, Searle's Chinese-room example really is a thought experiment. No human who was truly ignorant of Chinese could really manipulate Chinese symbols in the way Searle describes in real time and pass the Turing test. From the point of view of the person in the room, of course, there is no understanding of Chinese. But the real issue is whether the whole system, including the person in the room, the English instructions, the script, the I/O mechanisms, etc., understands. Searle denies that it does or that any computer system with similar capabilities understands Chinese, but his basis for

such denials is unconvincing. If neurons can be combined to form a complex system that produces understanding, why cannot electronic components be combined to produce understanding? Searle's central argument is that digital electronic computers cannot have semantics (qv). But, if biological brains have both syntax and semantics with the right causal powers why cannot electronic brains be so endowed?

Third, computer simulation of an activity does not rule out computer duplication of the activity. A computer that simulates flight does not fly a plane. Such a computer need not leave the ground to create a good simulation. But a computer in a plane that guides the craft in flight is not simulating flying a plane; it is flying a plane. Computers today only simulate certain aspects of thinking and would not pass the Turing test. But, if the linguistic abilities of computers are greatly enhanced to rival some machines in fiction, such as HAL and C<sub>3</sub>PO, what would be the ground or the point of insisting that such machines simulate but do not duplicate thinking?

### Legacy of the Turing Test

Disagreement about how much weight to attach to a test is not uncommon in science, and disagreement about the significance of the Turing test is certainly not surprising given the controversial nature of the subject matter. Both proponents and critics of the Turing test have insights to offer. Suppose that sometime in the future a computer did pass the Turing test. Surely, this would provide significant evidence that a machine thinks. Yet, lingering doubts, natural curiosity, and good scientific practice would lead to further investigation. Can the computer pass more Turing tests? (Too easy?) What else can it do? (Too narrow?) How does it do it? (Too shallow?) Indeed, Turing might not be dissatisfied with this outcome, for his own goal after all was to focus questions on his test and away from the original question "Can machines think?"

**Restricted Turing Tests.** In addition to a particular test, Turing's approach has served as an inspiration for the methodology and goals of AI. For example, Turing-like tests have been useful in evaluating and justifying expert systems (qv) (21). In a restricted Turing test judges compare computer behavior with human behavior in a narrow range of activity. Judges may be asked to determine which results were generated by a computer and which by a human or they may be asked simply to rank the results along some dimension. For instance, MYCIN, a program that diagnoses and recommends treatment for meningitis, was evaluated by a panel of expert judges comparing MYCIN's analyses of a series of cases with the analyses of physicians of various levels of training and experience. In this test the judges rated MYCIN's analyses to be equivalent or preferable to those of actual physicians (22).

Restricted Turing-test methodology also has been useful in probing the strengths and weaknesses of computer models of human behavior. For instance, different versions of PARRY, a program that simulates paranoid behavior, have been evaluated by several restricted Turing tests (23). In one test psychiatrists interviewed both an actual patient and PARRY over a teletype system, and in another test transcripts of interviews with PARRY as well as transcripts of interviews with actual patients were sent to psychiatrists and to computer scientists for discrimination and evaluation. In general, PARRY was

identified as the real paranoid patient about half the time. But as Colby (24) stresses, the valuable information from these tests for improving the model comes from the ratings of various dimensions of the model, not from asking the more general man-machine question.

### Conclusion

The Turing test symbolizes a long-range goal of AI, i.e., the creation of a computer with general intelligence. Turing knew the best path to creating a computer that would pass the Turing test is not to program a machine with fixed knowledge but to build a child machine that can be educated. Presumably, this computer would learn from experience and would use natural language to increase its knowledge. This computer would demonstrate a practical intelligence when interacting with the everyday world. It would solve its own problems and accomplish its own goals. In effect, this computer would do just what Descartes thought a machine can never do. Whether such a computer will ever exist is an open question; but if it did, who would deny it thought?

### BIBLIOGRAPHY

1. R. Descartes, *Discourse on Method*, (1637). Published in *The Philosophical Works of Descartes*, trans. by E. S. Haldane and G. R. T. Ross, Cambridge University Press, Cambridge, UK, 1973.
2. A. M. Turing, "Computing machinery and intelligence," *Mind* **59**, 433-460 (1950).
3. Reference 2, p. 19.
4. P. H. Millar, "On the point of the imitation game," *Mind* **82**, 595 (1973).
5. Reference 18, p. 423.
6. K. Gunderson, *Mentality and Machines*, Doubleday, Garden City, NY, 1971.
7. Reference 6, p. 44.
8. J. H. Moor, "An analysis of the Turing test," *Philos. Stud.* **30**, 249-257 (1976).
9. D. R. Hofstadter, "Metamagical themas: A coffeehouse conversation on the Turing test to determine if a machine can think," *Sci. Am.* **244**, 15-36 (1981).
10. N. Block, "Psychologism and behaviorism," *Philos. Rev.* **40**, 5-43 (1981).
11. J. Weizenbaum, *Computer Power and Human Reason*, W. H. Freeman, San Francisco, CA, pp. 1-16, 1976.
12. Reference 11, p. 6.
13. Reference 11, p. 7.
14. Reference 10, p. 10.
15. Reference 6, pp. 53-55.
16. J. Fodor, *Psychological Explanation*, Random House, New York, pp. 126-127, 1968.
17. D. F. Stalker, "Why machines can't think: A reply to James Moor," *Philos. Stud.* **34**, 317-320 (1978).
18. J. R. Searle, "Minds, brains, and programs," *Behav. Brain Sci.* **3**, 417-457 (1980).
19. Reference 18, p. 419.
20. J. R. Searle, Can Computers Think?, in *Minds, Brains and Science*, Harvard University Press, Cambridge, MA, p. 34, 1984.
21. R. O. Duda and E. H. Shortliffe, "Expert systems research," *Science* **220**, 261-268 (1983).
22. V. L. Yu, L. M. Fagan, S. M. Wraith, W. J. Clancey, A. C. Scott, J. Hannigan, R. L. Blum, B. G. Buchanan, and S. N. Cohen, "Antimicrobial selection by a computer," *J. Am. Med. Assoc.* **242**, 1279-1282 (1979).
23. K. M. Colby, F. D. Hilf, S. Weber, and H. C. Kraemer, "Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes," *Artif. Intell.* **3**, 199-221 (1972).
24. K. M. Colby, "Modeling a paranoid mind," *Behav. Brain Sci.* **4**, 515-560 (1981).

J. H. MOOR  
Dartmouth College

## U

**UNCERTAINTY AND PROBABILITY IN AI.** See Medical-advice systems; Reasoning, plausible.

**USER INTERFACE.** See Human-computer interaction.



**VERSION SPACES.** See Concept learning; Learning.

## VISION, EARLY

The study of biological perception has had an enormous influence on the development of computational vision. Unfortunately, the most common observation about biological—in particular, human—vision is its immediacy: simply open your eyes and a percept of the world appears. This apparently effortless speed implies to many that vision is relatively simple and that general-purpose vision systems should not be too difficult to construct. However, this effortless speed results not from the simplicity of vision but rather from the immense amount of specialized wetware, the biological equivalent of hardware, dedicated to it. In humans the visual system occupies a major portion of the cortex. Vision is, in fact, immensely complex, and it has turned out to be immensely difficult to construct successful vision systems. Endless applications in areas as diverse as robotics (qv), biomedicine, and remote sensing all support this conclusion.

Two basic problems confront the designers of complex early vision systems: What are the fundamental pieces, or the individual tasks comprising vision, and how should they be solved. The problems are clearly related; either the designer has a problem for which he can foresee a solution or he has a solution “in search of a problem.” Although much of the insight into how to decompose vision has come from mathematics, physics, and computer science/engineering, perhaps the most powerful influence to date has been from the study of biological vision systems. This entry is a historical survey of the modern development of early computational vision. The goal is to illustrate how diverse the influences on computational vision have been and to argue that such diversity is necessary.

The entry proceeds as follows. It begins about 100 years ago with the two great vision scientists von Helmholtz (1821–1894) and Mach (1838–1916), see General References. Two themes emerge from their different views of vision. In Helmholtz a clear separation can be seen between low-level and high-level processing, or what is now sometimes called early and later processing; and in Mach there is a separation between the analysis of a task and the mechanism proposed to accomplish it. (Early processing does not imply a temporal dimension; rather, the term “early” denotes processing from the retina back into the cortex, and “later” denotes the latter stages of cortical processing). These two themes were present in the first attempt at a complete computer vision system—the one by L. Roberts—and they persist to the present. In Roberts’s system there was a clear separation between low-level processing, or the extraction of a cartoonlike line drawing out of an image, and high-level processing, or the recognition of objects. And the mechanisms applied at these levels depended on the tasks; the low-level mechanism being one of so-called edge detection (qv) and the higher level one of object matching into a database. Modern computational theories, such as the one proposed by Marr (1), postulate more elaborate interfaces: “primal sketches.” Although this thread is common, the main

evolution of computational vision has been an appreciation of the immense complexity involved in both of these stages, with one paradigm after another attempting to grapple with it. Different paradigms have arisen for low- and high-level processing, and some have even emerged for intermediate stages. Strong forms of so-called inverse optics, pieced together with little or no interaction, have now given way to an increased appreciation of abstract structure. That is, it has now become clear that it is essentially impossible to exactly invert the scene projection process; rather, the search is on for discovering which aspects of the structure of the world can—and should—be recovered.

The detailed evolution of the field can be thought of in terms of two pendula, one representing the tension between low-level and high-level vision, and the other between the formulation of the task and the techniques chosen to solve it. This is very much a personal view, as are the examples that I have chosen to illustrate how these pendula swing back and forth in time. Interestingly, early on in the development of the field they were assumed to be rather separate from one another, but as the field began to mature, their interrelationships became more clear as well. The tension between low and high level vision developed into a concern for the type of knowledge to be applied, the specifics of which are clearly related to both task formulation and technique employed.

The vision problem can be summarized as follows. Three-dimensional physical structure in the scene projects into two-dimensional structure in the image. This process must be inverted; i.e., somehow, physical structures must be inferred from image structures. For each class of related physical and image structures a microinverse problem can be formulated, and many such problems exist, as we shall describe. Early (low level) vision consists of those problems for which the solution is driven by general-purpose assumptions and special-purpose hardware, whereas later (high level) vision consists of those problems for which the solution is driven by special-purpose assumptions and general-purpose hardware. Or stated differently, in early vision, if something is understood about structure (of the world), something can be inferred about function in the visual system; whereas in later vision it appears that function must be understood before structure.

The focus of this entry is on the evolution of ideas rather than on algorithms. There is more concentration on the classical foundations of the field than on current approaches. Several other articles in this *Encyclopedia* address these different aspects of vision in detail, and cross-references to them are indicated whenever possible. Furthermore, given space limitations the entry needs to be somewhat selective about material. Many book length treatments of computational vision (2–8), image processing (9,10), and visual perception (11–16) are available and should be consulted along with this entry. It is also worthwhile to consult the annual list of publications in computer vision and image processing compiled every year and published by Rosenfeld in the journal *Computer Vision, Graphics, and Image Processing*. Also, several recent collections have emphasized the relationships between biological and computational vision (17,18).

### First Paradigm: Segmentation in Low Level and High Level Vision

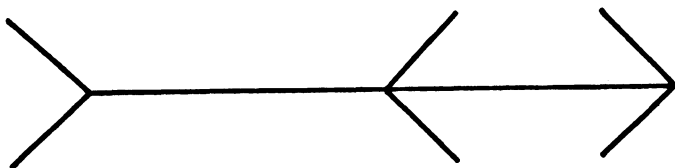
The study of vision begins with the grand decomposition suggested by Hermann von Helmholtz (19).

**Helmholtz: Physiological Optics and Unconscious Inference.** In his treatise on physiological optics (19) Helmholtz sketched a theory of vision in which the eye acted as a transducer of light into the nervous system, which then performed "unconscious inferences" in order to compose internal versions of percepts. That is, he asserted that there was a low-level component to vision, dominated by physics and physical models, and a high level component in which the inferences (qv) took place. Unfortunately, the only language that he had for talking about inferences was the rather loose one of what he took to be "conscious inferences," or the logic of premises and conclusions. High level vision is, he therefore asserted, the same sort of activity as is normally involved in cognition and thinking, although one is unaware of it.

Although Helmholtz was (unfortunately) rather vague about unconscious inferences, his studies of early vision are still remarkably fresh and insightful. To illustrate, consider his study of the transduction properties of the eye. Perhaps inspired by his work in physics, he countered a rather widespread belief that the eye was a "perfect" optical instrument by actually measuring its optical properties. He observed, as is commonly known today, that the eye is far from perfect (20). It exhibits the many different forms of aberration and distortion to which all physically realized systems are susceptible.

The result of such optical imperfections in the eye is that images do not fall on the retina in perfect focus but are blurred regardless of how well the lens is functioning. Helmholtz looked for perceptual consequences of such blurring and found many, one of which he believed to be the Mueller-Lyer illusion (see Fig. 1). His reasoning was as follows. On a figure such as the Mueller-Lyer, the areas between the lines forming the acute angles will be blurred more than the areas within the obtuse ones, thereby stretching the lines into the acute angles more than the obtuse ones. Such a distortion is precisely in the direction of the illusion and was, for Helmholtz, its causal explanation.

Such is visual theorizing of the best sort. A task is posed (what are the optical properties of the eye?) and solved in a theoretical fashion that is consistent with empirical data (the spherical aberration was actually measured). Finally, the the-



**Figure 1.** Mueller-Lyer illusion. Although the arrowheads delimit two line segments of equal length, the one enclosed in the convex region appears shorter than the one in the concave region. Helmholtz believed that this is because people's visual systems somehow fill in" the convex portion more than the concave portion, thereby effecting the length judgments. Gregory (12), on the other hand, believed that interpretations of these line segments as projections of 3-D structures lies at the foundation of the illusion. Whatever the mechanism responsible, however, such illusions indicate that everyone's perception of structure in the world is not veridical but rather depends on contextual influences from many possible sources.

ory was applied to explain observed phenomena (such as the Mueller-Lyer illusion).

Helmholtz was correct in observing that the eye is an imperfect optical instrument. But he was mostly wrong in that his explanation of the Mueller-Lyer illusion cannot account for the entire effect. This has been determined only recently using an elaborate optical technique to project a highly focused image onto the retina (21). Such techniques indicate that optical blurring can account for at most 15% of the illusory effect. Nevertheless, considerations of the physics inherent in the imaging process will emerge in different ways as a major theme in computational vision.

**Roberts's System: Segmentation and Matching.** Inherent in Helmholtz's theory is a distinction between the types of processing that take place early on and then later in the visual process. Such a distinction lies at the basis of modern computational theories as well, beginning with the first real attempt to design a full system. In a seminal thesis Roberts (22) described what is probably the first computer-vision system. Although it is not clear that he was directly influenced by Helmholtz, he, too, decomposed processing into a low level stage, in which a line drawing was abstracted out of an image, and a high level stage, in which the line drawing was matched against a universe of prototypes. Thus the physically motivated processing was concentrated on the extraction of the line drawing, and "unconscious inferences" were used to match it into a database of objects.

In order to effect this matching, it was necessary for Roberts to restrict the possible universe of objects that his system could encounter. He worked in a miniworld of polyhedral objects composed entirely of blocks—the so-called blocks world—a class of assumptions that influenced computational vision for more than a decade. Even though the universe of objects was simple, however, it did not follow that the matching would be trivial; in the process various transformation parameters such as complex object decomposition, depth, and rotation had to be computed.

The early portion of Roberts's system was concerned with the problem of edge detection (qv), or the identification of those positions in images that indicate interesting physical events. The locus of these positions then comprised a line drawing. The motivation behind "line drawings" can be seen intuitively in cartoons, or drawings which are, in some sense, equivalent to full images. That is, both convey sufficient information to satisfy one's high-level, inferential processes. More specifically, the line drawing was taken to represent a segmentation of the image into meaningful pieces, each of which was taken to be the projection of a meaningful portion of a physical object. The outlines of these pieces then comprised the line drawing.

Roberts's approach to edge detection was based on the observation that distinct physical events (say, the sides of a cube) give rise to distinct image events (intensities). And the image locations at which these events meet have special significance: they are the edges of the cube. Thus, if the points of intensity change could be located and joined, the result would be a perfect line drawing of a projected cube.

To locate the edges of the cube, observe further that intensity changes rapidly there. Calculus tells one that, for smooth functions, rapid changes in the value of the function at a point are accompanied by large values in the derivative of the function at that point. Hence Roberts derived a discrete approxi-

mation to a gradient operator elegantly simple in computational form. His plan was to convolve this operator against the image, and then to select the strongest of these convolution values by thresholding. Finally, a linking process would select high gradient "edge" points to be fit by straight lines. Note how a distinct higher level constraint enters Roberts's formulation at this point: the straight sides of his scene polyhedra project into straight "edges" in the image. Helmholtz made similar observations about straight and converging lines. There is further discussion about the influence of such "high-level" or domain-specific knowledge (qv) on early processing later in the entry.

However, as described below, there is a lot more to low-level, early processing than was thought at this time. Roberts was never actually able to get a perfect line drawing from this early processor, and there is a basic sense in which the perfect line drawing is still elusive. But this is a fascinating story in itself since what began as the pursuit of the perfect line drawing has evolved into the full study of early vision. Again the entry starts historically, this time with Helmholtz's contemporary E. Mach.

**Similarities versus Differences.** Before beginning the discussion of edge detection, however, consider a tangential point. Segmentation (qv) can be approached in two different ways: either by searching for differences, or points along segmentation boundaries or by searching for similarities, or points, within segmentation regions. Edge detection is the approach for determining which points are different, and it is the approach adopted by Roberts. Region growing (see Region-based segmentation) is another approach to segmentation designed for determining which points are the same (8). The rationale for region growing was that, since differentiation emphasizes noise, segmentations might be found more reliably by smoothing "within" regions rather than differentiating between them.

However, it should be stressed that the duality of edge detection and region growing does not imply that one or the other is unnecessary. Rather, they are complementary and almost always work together. Consider Roberts's system again; note that after edge detection (differentiation), similar points (i.e., the locations at which the differential convolution survived thresholding) are linked into lines; this linking process is a kind of one-dimensional region-growing process. That is, points are defined to be similar if they are associated with a common straight-line segment in the least-mean-square sense. Thus, following linking, it is as if all of the boundary points had been "smoothed" into a bounding contour since their (prethresholding) individual differences are now gone. The key information provided by the discontinuity measurement—the edge operator—has now been summarized into a more global, abstract form. This complementarity emerges again and again throughout the evolution of early vision; eventually it emerges as grouping. Consider now the edge-detection route, however, since this was by far the most prominent of the two.

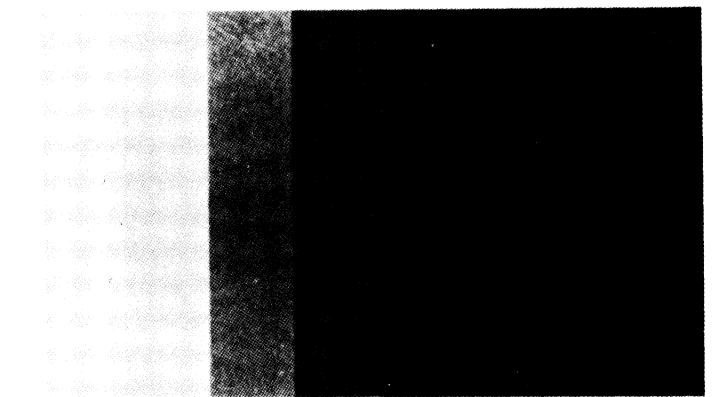
**Laplacians, Mach, and Edges.** The modern study of edges has its roots in the studies of E. Mach. To contrast him with his contemporary Helmholtz, Mach was interested in image sharpening rather than blurring, and in explanations couched in terms of neural networks rather than in physiological optics. The phenomenon of sharpening is known as Mach bands,

or the addition of subjective bright and dark lines (bands) on either side of an intensity change (see Fig. 2). Such bands indicate that the "eye" (i.e. the visual system) is sensitive not only to image intensities but also to their (first and second) derivatives.

Mach bands give a clear indication that the subjective impression of brightness and of contrast is highly dependent on spatial context. That is, one's impressions of brightness and of contrast are not isomorphic with the intensity of light impinging on the retinas but rather are derived—or computed—from it.

How can these computations be understood? Mach believed that psychophysical laws, such as the ones underlying brightness and contrast phenomena, had their proper explanation in terms of properties of neural networks, not in terms of pure physics or purely "physical events." The particulars of Mach's explanation were posed mathematically in terms of "a reciprocal interaction of neighboring areas of the retina" (23). He formulated mathematical relationships involving the Laplacian operator, a symmetric second differential of the image intensities (see below). He cited (then) current neuroanatomical data by Ritter (23) that postulated a regular arrangement of cells on the retina and characterized the function of these cells mathematically. And he postulated that the result of the neural interactions between these cells was a "sensation surface" on which the brightness effects were present. Thus, Mach, in discussing such surfaces, was talking directly about representations (read: *re*-presentations); he was concerned with possible constraints from the "wetware."

**Lateral Inhibition: From Operators to Cooperative Computation.** Although Mach was able to infer the nature of processing taking place immediately after the retina, it was not until a revolutionary innovation in neurophysiology—the development of microelectrodes for single-cell recording—that his inferences could be verified experimentally. This was first done in the eye of the horseshoe crab *limulus* and has led to much more accurate mathematical models. Such models are said to exhibit lateral inhibition, or a regular structure in which the response at a particular retinal point is derived from excit-



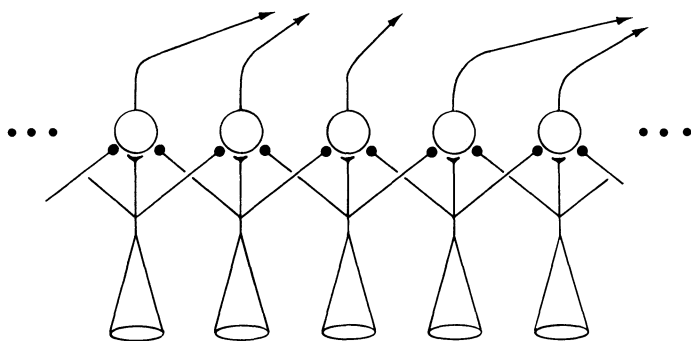
**Figure 2.** Illustration of Mach bands (11). The image consists of a sequence of rectangular regions of constant intensity, the "edges" between regions are therefore perfect "step edges." However, the intensity does not appear constant to a viewer. On either side of each edge are two Mach bands, a darker one (on the dark side of the edge) and a lighter one on the other side. Again, these bands indicate that what one sees is not "what's out there" but rather is a context-dependent computation driven by it together with additional constraints.

atory contributions at that point together with inhibitory interactions from neighboring points (11,24) (see Fig. 3). Notice, in particular, the regular neural architecture for implementing lateral inhibition, in which the same local structure is repeated across the spatial array. Viewed spatially, the lateral inhibitory structure looks circularly symmetric, with an excitatory central area enclosed within a negative, or inhibitory, surround. Or, in other words, the response at a retinal point is a function of the context around that point.

An essential aspect of this context is the presence of intensity changes in the visual array. As said in the discussion of Roberts's system, such changes are important because they often indicate the presence of physical object contours. In fact, the functional significance of Mach bands has often been attributed to their edge-enhancement effect: if one is to navigate through the physical world on the basis of sensory information, one certainly needs to locate object contours.

Lateral excitatory and inhibitory networks are without doubt one of the most ubiquitous mechanisms in biological vision systems. Lateral inhibitory networks play a clear role in regulating the dynamic range of the eye (25) and otherwise performing a sort of local sharpening, or maxima selection, at the neural level (11). But more generally they have led to a general view of the kind of computational architecture that should be employed in early vision, an architecture of regularly interconnected networks of rather simple processors. But before these networks are developed, consider the local view of lateral-inhibitory-based edge operators.

**Discontinuity and Edge Detection.** The classical approach to edge detection is differentiation. This is typically accomplished in two stages: the convolution of an operator against the image and some process for interpretation of the operator's responses. Or stated in more general terms, the stages consist of a measurement process followed by a detection process. It is noted above that there is a basic sense in which the two are complementary: if edge detection is a differential process, interpretation must be an integrative one (recall Region growing). Note that there are two convergent approaches to the design of measurement operators, either as numerical approximations to derivatives of different order or as inferences about how primates might do it.



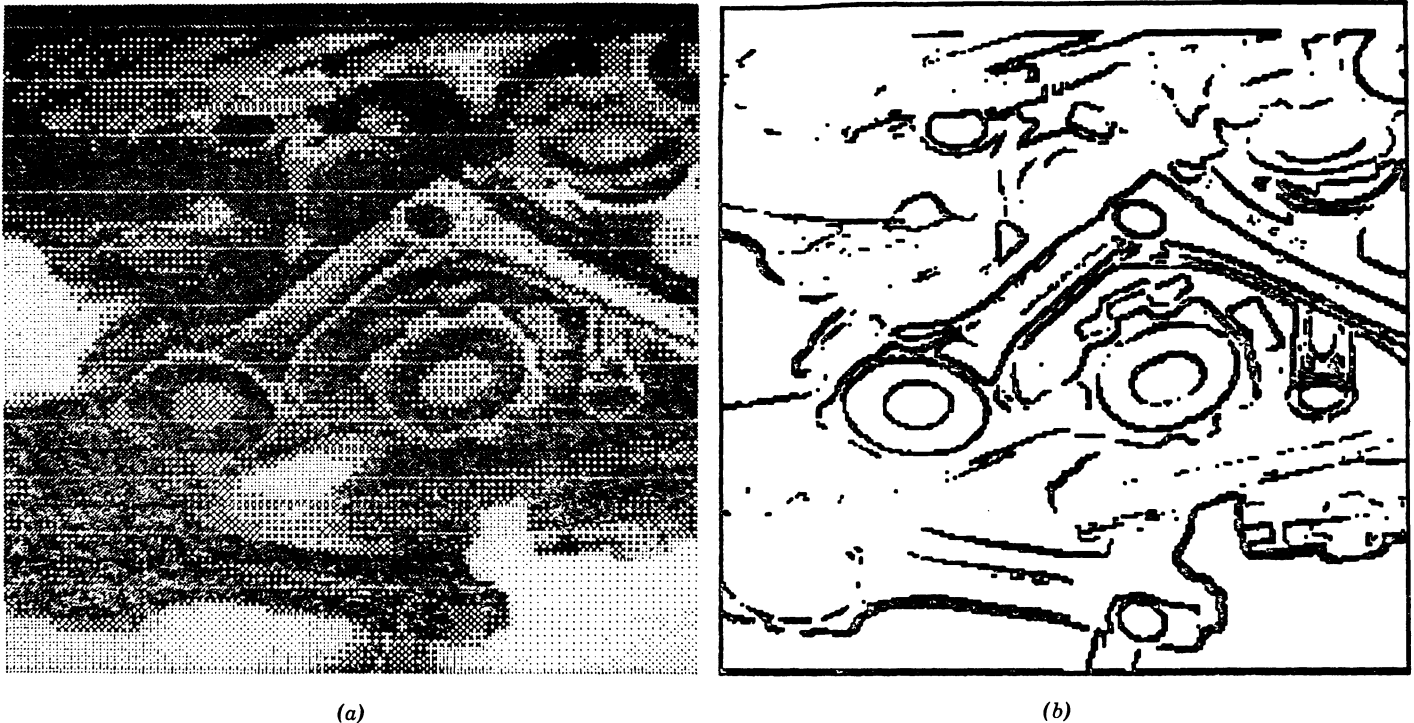
**Figure 3.** Example of the computational structure of lateral inhibition. In this diagram the cones represent light receptors, and the large circles represent "summation" devices. Note how the level of activity in each receptor contributes positively to the result (excites the summation device), whereas the level of activity in neighboring receptors inhibits it (filled circles). In general, the architecture is one of arrays of units with near-neighbor interactions, both excitatory and inhibitory, with the local computation a simple one.

**Edge detection as Differentiation.** If the surfaces of physical objects project different image intensities, it would seem that the locations of the physical object "edges" could be inferred from the places where intensity changes rapidly. These rapid intensity changes can be detected, under the assumption that the world projects smooth image-intensity functions to which differential calculus applies, by locating those positions at which the first derivative (spatial gradient) is high; or where the second derivative crosses zero (this assumption, and the approach that it implies, is questioned below). However, there are important numerical issues to be confronted as well, so that the estimates of the derivatives are as accurate as they can be. Trade-offs between these two issues—differentiation and numerical stability—are classical. They led to better approximations to the gradient than the Roberts operator [see the Sobel operator in Duda and Hart (3), as well as Kirsch (26); the numerical issues are discussed in Hildebrandt (27) and implications for edge detection are in many textbooks (2,4,8). Before proceeding, it should be noted that in spite of the predominant identification of edge detection with differentiation, other approaches emerged. Chief among these were a formulation of edge detection as hypothesis testing, so that both the differences in edge profiles and their inherent noisy variation could be taken into account (5,28,29), and an observation that fitting "surfaces" to intensity distributions was a more numerically stable way of finding step discontinuities (30–32). But they still did not work sufficiently well (see Fig. 4).

Shown below is the evidence for the second major influence: early primate vision.

**Shape of Visual Receptive Fields.** A virtual revolution in the understanding of early visual physiology took place from single-cell recordings in cat and monkey visual systems. The receptive field of a cell indicates how arrangements of light stimuli will effect its activity; in effect, the receptive field characterizes aspects of what the neuron is doing. Hubel and Wiesel (33) discovered a striking arrangement in receptive field structure. Their discovery can be appreciated as follows. Suppose an electrode is indicating the level of activity (firing rate) of a neuron in the visual system. If a spot of light is shone somewhere on the retina, processing may percolate back to influence the firing of that cell. This will be true for some locations in the retinal array, and it may be either excitatory—leading to an increase in the firing rate—or inhibitory, leading to a decrease (below some "resting" or spontaneous level). The shapes of these receptive fields in retinal ganglion cells resemble a circular-surround organization that has been modeled as a difference of two Gaussians (34,35) (see Fig. 5). And they come in two flavors: excitatory center with inhibitory surround, and inhibitory center with excitatory surround.

In the cortex, however, the structure of receptive fields changes dramatically. Here they exhibit the additional property of being orientation selective. That is, individual cells respond better to lines and intensity edges than to isolated points. And their response varies as a function of the orientation of the lines and points. The receptive fields are elongated (see Fig. 6). The temptation to identify them with operators for edge and line detection is overwhelming, and this is normally done. However, at this time identification was almost purely local, with little consideration of the network interaction necessarily taking place between these local pieces. The only interactions considered were those required for constructing hierarchies of cells as building blocks to more complex functionality.



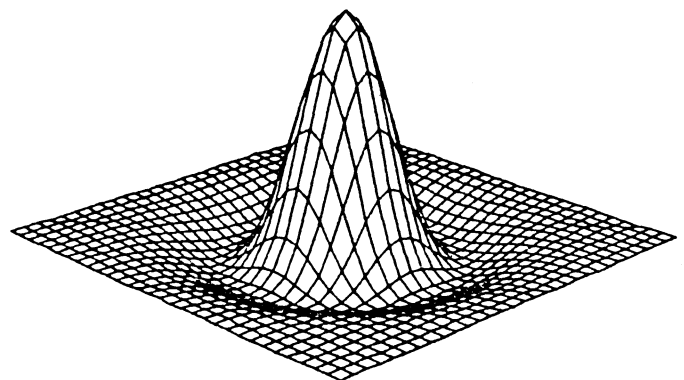
**Figure 4.** Illustration of the Sobel edge detector. (a) Original image of automotive parts ( $256 \times 256$ ) pixel resolution. For display this image has been quantized to six gray levels. (b) Binary image indicating the positions at which the maximal Sobel responses were located. Thresholds were specified by a locally adaptive algorithm. Note that although some of the prominent edges have been found, it is certainly not the case that all have been found. Problems clearly arise in image areas that do not contain steplike edge changes.

**Edge Detection and Scale.** Visual receptive fields span a range of sizes, a point that is loosely consistent with quite a bit of psychophysics regarding threshold perception. Consider, e.g., a display consisting of a sinusoidal grating. The minimal contrast necessary to see the grating is a function of its spatial frequency (11). Wilson and Bergen (36) have empirically determined that these psychophysical data are consistent with four separate channels of processing. Although these channels have not yet been related quantitatively back to the physiology, they seem to indicate (at least abstractly) a number of parallel functional streams. But this point has always been puzzling to computational modelers. If the receptive fields of these cells participate in edge detection, why the variability in scale? What role does scale play in edge detection? Several suggestions have emerged. First, given the noise problems inherent in early vision, from quantization, occlusion, and receptor processes, some sort of averaging would seem necessary to reduce it. For example, if one wished to measure a local feature reliably, say an edge configuration, increasing the size of the operators could lead to increased performance (equivalent detectability with decreasing signal-to-noise ratio) (37). However, there must be more to it than this because larger operators will cover more of the image; hence they may cover more structure than, say, one edge.

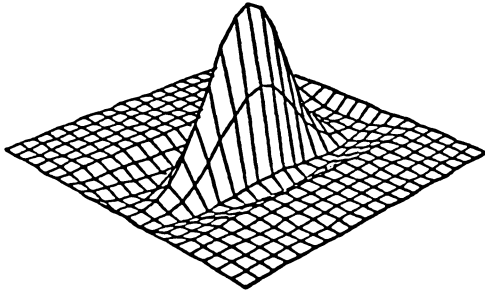
In addition to numerical issues, observe that structure in the world arises at different scales as well. Observe, in particular, that certain physical events are highly localized in space (say, the locus of points along which the faces of a cube meet), whereas others are much less localized; they span more space (say, the locus of points defining an animal's limb). These observations indicate the scale at which different physical events

are taking place. Land (38) observed, e.g., that changes in physical objects are usually highly localized (say, at the occluding edge between them), whereas changes in lighting are typically much more diffuse. The scale of intensity events thus purportedly "decomposes" lighting from reflectance. Witkin (39) has suggested a scale space (qv) for studying events at these different scales (see also General-Purpose Models Revisited). Could it be that the variation in receptive field size is tuned to events of different "scales" in the world?

**The Elusive Edge Operator.** Marr and Hildreth (40) tried to link the above notion of scale with specific ideas for edge detection. Selecting from the above facts, they observed that the circular surround operators could be approximated mathematically



**Figure 5.** Illustration of Laplacian of a Gaussian edge operator. The illustration also approximates the difference-of-Gaussian receptive fields typical of those found in primate retina.



**Figure 6.** Type of simple cell receptive field of the sort that could be found in early primate visual cortex. Note that it resembles the circular-surround retinal receptive field in Figure 5, although now it is elongated. Such elongation illustrates an orientation preference. In computer vision such operators are known as “line detectors.”

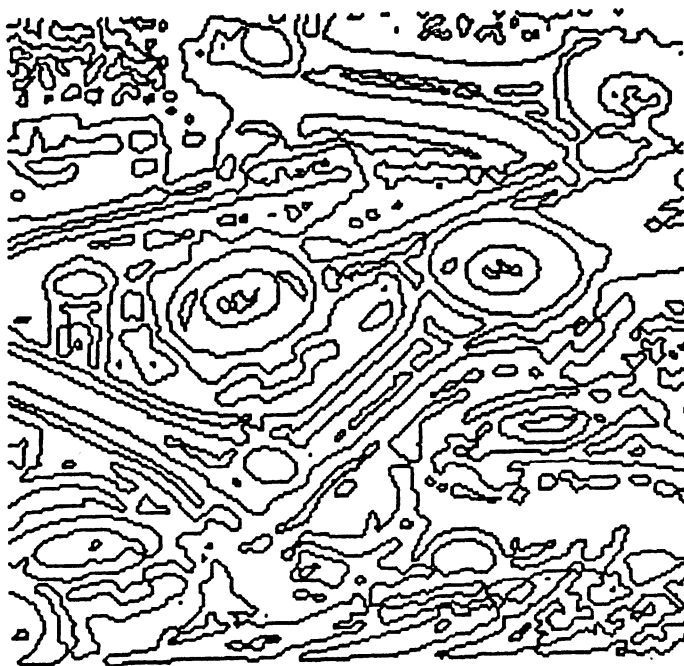
ically as  $\nabla^2 G$ , the Laplacian of a Gaussian; that since the Laplacian is a second-derivative operator (recall Mach (23)), step changes in intensity can be localized by its zero crossings; and that the different sizes of operators would be sensitive to events (i.e., “edges”) at different spatial scales (see Edge detection).

Such is a wonderful confluence of events. The problem of edge detection, with which researchers in computational vision have been preoccupied for two decades, could now be solved in a way that is consistent with psychophysical and neurophysiological data and, moreover, provides a functional explanation for it. Unfortunately, however, the scheme cannot work in general for two reasons. First, implicit in it (and in the design of most other edge operators) is an assumption that the intensity structure is a step function across the edge and that

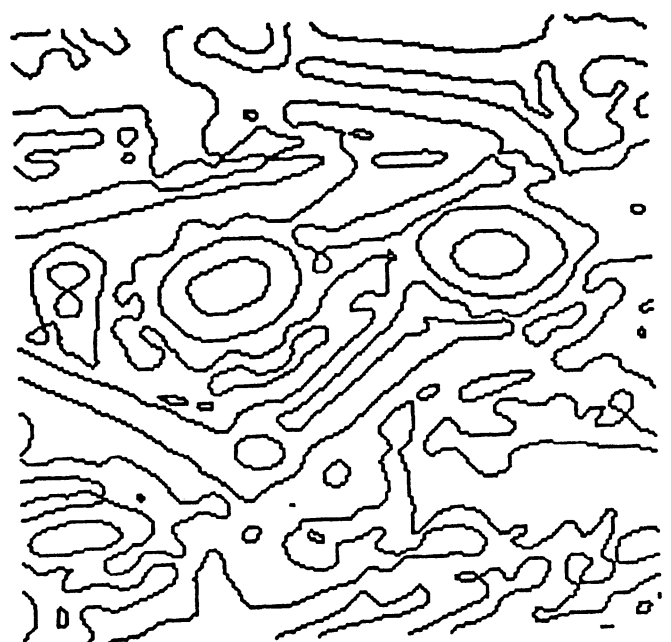
the curvature of the edge is zero; i.e., that the edge is straight. It can be shown that these are precisely the conditions under which this operator works successfully but that few edges in natural scenes are of this form (see Fig. 7). Rather, there is a plethora of different physical configurations that can give rise to edges, and these must be taken into account (41) (see also From Structure into Function, below). Moreover, consequent psychophysical predictions from the Marr–Hildreth operator have not been supported (42). The perfect edge operator remains elusive.

**Image Representation and Communication.** If the circular-surround receptive fields—especially those in the retina and the lateral geniculate nucleus (LGN)—are not involved in edge detection, what other function might they be accomplishing? Since the retina is “an outgrowth of the brain,” the problem of communicating image information from the retina to the cortex stably and reliably arises. Laughlin et al. (37) have shown that linear predictive coding theory leads to a model that fits these receptive fields strikingly well (at least for certain animals), and considerations of numerical stability lead to the separation of opposite contrast data (43). It thus would seem that edge detection is likely to begin in the cortex, which opens up the door for much more complex processing. The classical idea of a single edge-detection operator seems unlikely; it remains necessary to discover the mappings between physical scene structure and images and between image structure and visual function.

**Naive Physiology: Hierarchies of Feature Detectors.** Although lurking in the background, the influence of (then) current notions in physiology on computer vision has always been quite strong. The basic model was feature detection (see Fea-



(a)



(b)

**Figure 7.** Edge locations (zero crossings) obtained with the operator in Figure 5. (b) Compare with Figure 4b. Note that the zero crossings form closed contours, although these sometimes have little connection with the physical objects comprising the scene. Two size operators are shown [a small one in (a) and a large one in (b)] in order to illustrate that the problem is not simply one of “scale.” Neither one is completely satisfactory for locating edges.



ture extraction), a two-stage procedure in which operators were first convolved against the image, and then the best match (i.e., the strongest convolution) was selected by a process of thresholding. The operators were somehow matched to the image projections of certain stimuli in the world, following the ideas presented in the classic paper by Lettvin, Maturana, McCulloch, and Pitts (44) in which circular-surround receptive fields most sensitive to movement of spots of a particular size and velocity were interpreted as bug detectors. In addition to this general perspective, the model of physiology that was most strongly influencing researchers in computer vision was the hierarchical one put forth by Hubel and Wiesel beginning in 1962 [see the review in the paper by Hubel and Wiesel (33)] in which visual neurons exhibited three types of receptive field structures: simple, complex, and hypercomplex. Simple cells were defined as those in which the subdomains were linear and separable (hence, simple to characterize); complex cells as those in which the subdomains were nonlinear and overlapping (and hence complex to characterize); and hypercomplex cells, which were the most difficult of all to capture. Hubel and Wiesel further hypothesized a hierarchical relationship between cells: retinal ganglion cells fed into the LGN, maintaining the circular-surround receptive fields discussed above. These LGN cells are then combined into elongated simple cells, which are then combined successively into complex and hypercomplex cells. It was loosely asserted that simple cells are involved in edge and curve detection and hypercomplex cells in detection of "higher order" properties such as "corners" or "end points," although even at this time problems were surfacing. How, e.g., could such cells detect dashed curves in noise (45)? More precisely, the receptive fields were modeled as operators, and the question became: how could operator convolutions followed by thresholding detect dashed curves in noise? No clear function was proposed for complex cells, and (as shown below, the wonderful simplicity of hierarchical arrangements of feature detectors gives way to more realistic computations. There is more to "bug" detection in the frog than circular-surround receptive fields, and there is more to "edge" detection in humans than individual simple cells.

**Knowledge in the Edge-Detection Process.** As the measurements of image intensity changes evolved (recall the "edge-detection" operators), so have the processes for interpreting, or selecting, a "truest" one from among them. Such selection is necessary because the value of the convolution is nonzero almost everywhere due to microstructure and noise. Somehow the significant responses must be separated from the insignificant ones.

**Thresholding, Local Maxima Selection, and Hough Transforms.** In the simplest case normal thresholding suffices: Simply select the strongest (highest value) convolutions; all others are discounted. The idea behind thresholding is that true edges will project to real intensity differences and hence will lead to high convolution values. This holds for nonoriented edge operators when the decision is, essentially, whether a particular image location is part of an edge. The case of oriented operators is just slightly more complicated since not only the presence of edges must be separated from their absence (the noise responses must be eliminated) but the correct orientation of the edge must be chosen as well. Postulating the orientation at a point to be the same as the orientation of the operator mask with the highest convolution value is, in a sense, the best match, and thresholding is one way to select it.

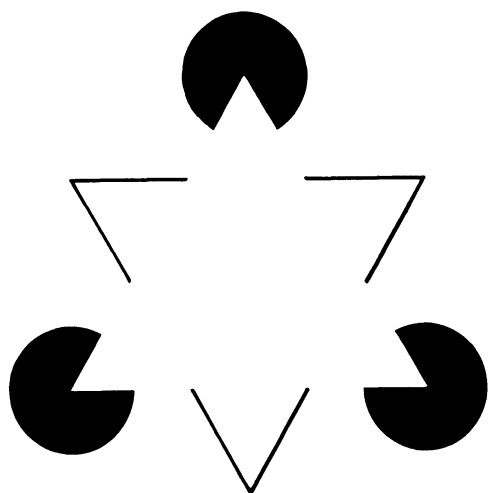
Thresholding can therefore be interpreted as a selection based on maxima in some first-order statistic, and the idea has evolved to include statistics of more general or less local features. Perhaps the first example in computer vision is the Hough transform (qv) (3), in which long straight lines are found by histogramming local estimates of their orientation and intercept. The Hough transform has been generalized and applied by Ballard (46), Davis (47), and others.

Thresholding can also be viewed as a mapping that takes an image (in which entries are the value of a particular convolution at a point) into another, binary image (in which the locations with values that survived thresholding have value 1 and all others have value 0). The recovery of more global structures thus requires further processing, say tracking along the 1s to find lines and contours.

**Vision as Controlled Hallucination.** Although the above arguments may seem persuasive at first, unfortunately thresholding's only virtue is its simplicity; it rarely works in practice. How can threshold values be selected? More than two decades of research have shown that, for any but the simplest of images and tasks, the knowledge required cannot be represented as threshold values (8). More sophisticated processing techniques must be employed, and a basic question arises regarding how to structure them. Two schools of thought have been prominent. Originally, it was believed that edge descriptions could somehow be extracted directly out of arbitrary images. This led to elaborate thresholding algorithms, thinning algorithms, etc. (3,8). But it is not at all clear that these mechanisms are sufficiently powerful to utilize the knowledge required for edge detection. It is very difficult to evaluate the results of individual algorithms out of any system's context, and early limitations in these algorithms, or apparent failures in larger contexts, led to the consideration of drastically different techniques. The suggestion that vision is, to a real extent, a process of "controlled hallucination" (48) emerged, the thrust of which implied that hypotheses about (or knowledge of) the object being imaged should directly influence the edge-extraction process. That is, knowledge from the highest, most abstract levels should influence the earliest, most primitive ones (e.g., Ref. 49). Key to this change in view is the observation that not only are intensity edges difficult to locate but many of the edges that define objects also have no image-intensity counterparts. It is possible to see edges where there is no change in intensity (50) (see Fig. 8).

**Top-Down versus Bottom-Up.** The question of which knowledge should be applied in the edge-detection process is a special case of a more general one: should image analysis be top-down or should it be bottom-up (see Processing, bottom-up and top-down)? Should it be data-driven or hypothesis-driven? Different schools emerged, with Rosenfeld and the image-processing community being associated with the bottom-up idea and much of the more traditional AI community associated with top-down approach. Perhaps the most prominent attempt at top-down edge detection was that of Shirai (51), in which a knowledge-based edge detector was designed to work only for images of cubes.

**Flow of Control in Knowledge-Based Edge Finding.** Shirai's system worked as follows. Standard image-differential techniques were used to locate a prominent intensity change. Because the universe of possible objects was limited, as in Roberts's case, to the blocks world, it could then be assumed that such a prominent edge point would be part of the bounding contour around the cube. After finding the orientation and



**Figure 8.** Kanisza subjective edge. Note how the apparent (bright) triangle is indicated by the missing corners in the dark circles and by the line terminations. Edges seem to be present even with no intensity changes.

length of this edge, it was then possible to hypothesize a putative cube model with certain size, etc., parameters instantiated to particular values. This model could then be used to predict the location of other putative edges, which could then be verified by looking at the image intensities in detail.

**Complexity of Edge Detection.** The use of knowledge all the way down to the lowest levels of the edge-finding process thus becomes quite complex, necessitating elaborate mechanisms for its control. Although this can lead to very high performance levels in restricted universes (such as the blocks micro-world), it also leads to brittle, highly specialized systems with little generality. Extending them to ever-so-slightly larger domains became arbitrarily difficult. In terms of object models, there is a formidable gap between the blocks world and the real world. This forced another look at complexity trade-offs and the flow of processing in vision systems. In retrospect it seems that too much was hoped for, with regard to the performance of local edge operators, a lesson that has still not clearly penetrated computational vision.

**Introduction of Surface Constraints.** Intermediate between knowledge about the exact object and about its edges is knowledge about how they fit together. In the blocks world, knowledge about edges is intimately linked to knowledge about surfaces, and other researchers began to introduce surface-intersection constraints directly into their programs as well. Mackworth (52), e.g., used the idea of gradient space (53–55), a representation of object-surface-normal properties (not image-intensity gradients) to determine which edge segments could physically belong together for polyhedral objects, i.e., gradient space makes explicit relations between the coordinates of polyhedral surface gradients and lines in an orthographically projected image (see also Refs. 56 and 57). Another intermediate step involved shape estimation, see Refs. 58 and 59.

**Rigidity of Early Systems.** The problem with these early systems was rigidity; The knowledge on which they were based—e.g., the Shirai constraints or the gradient-space constraints—were “hardwired” into the programs. They could work only for the idealized-object classes within which the constraints held.

Generalization was difficult, if not impossible. Historically it was time to back off from the detailed problems and to have a broader look, which is exactly what happened. As shown below, the network parallelism that was so obvious in early visual physiology now begins to play a much more prominent role. It suggests in particular a framework, a point of view toward vision, of how to derive and use general-purpose constraints from abstract assumptions about images and the world, assumptions that are far more realistic than those within the blocks world. The importance of intermediate levels of knowledge, such as that first suggested by gradient space, increases greatly, but its form is drastically different so that its use can become much more fluid and adaptable. For a further, in-depth discussion of the use of high-level knowledge in vision systems, see Tsotsos (60).

**Constraints and Assumptions.** There is often a minor point of confusion in the study of vision between the terms constraint and assumption. Assumptions about the universe, e.g., that it consists of flat surfaces, allow the derivation of constraints in algorithms, e.g., the equations for fitting planes rather than for fitting fifth-order polynomials. Clearly, the assumption of the blocks world leads to many such constraints, some of which have been described.

### Organization and Complexity

Two observations speak against the top-down, rigid analysis of images, and they are both related to complexity in a particular way (61). First, if one sees only what one expects to see, how can the visual system be responsive to unexpected events in the world? In the limit one would not even need to open his/her eyes! Second, on the basis of what trigger features—or database keys—are models selected? Somehow they must be derived from the image, and if the keys are just intensities, there is an immense complexity barrier to be overcome: there are an infinite number of different scenes that could project into any particular image. How can the correct one be selected in a reasonable amount of time?

The antidote to complexity is organization, and the answer to the complexity dilemma is classical. To illustrate, consider the Dewey Decimal Classification (Melvil Dewey, Lake Placid Education Foundation, 1922). If the books in a library were organized randomly, and there were  $n$  of them, it would take on average  $n/2$  examinations to find any particular one. But if the books were organized, say according to hierarchical categories as in the Dewey decimal system, the savings in search time could be enormous (under certain schemes search time can be shown to grow with  $\log(n)$  which, for large  $n$  is much slower than  $n/2$ ). Analogously, one needs to organize the knowledge in vision systems. The issue is not whether knowledge should be used, but how, what kind, and when it should be used. Intermediate, abstract (with respect to the models) knowledge must somehow be incorporated that captures the regularities of objects in the world. Of course, the earlier observation that physical edges project into image-intensity changes is one kind of knowledge, but this must be further generalized.

**Abstraction and General-Purpose Models.** Another demonstration can be invoked in support of intermediate levels of organization. Suppose you were looking at a totally unfamiliar scene, e.g., one from a scanning electron microscope or from an ultrasound scanner. Although it is unlikely that one would be

able to recognize the object—the scene can even be chosen so that it does not contain any real objects—one will still be able to describe what he/she sees. The description will be in general terms, perhaps involving geometric forms, apparent contours or corners, and will be of the sort that can be derived from any image. Such intermediate descriptions, and the knowledge incorporated into making (and interpreting) them, are what should be sought. What is needed are not assumptions about the entire universe of objects; as in the blocks world, by the time these assumptions give rise to useful constraints, they are too constraining. Rather, more abstract assumptions are needed about the intermediate kinds of structure that can arise in a wide class of natural scenes. These assumptions and the constraints they give rise to will provide the backbone for early processing.

**Ubiquity of Uncertainty.** Although the metaphor of libraries is instructive regarding the role of organization, it is misleading in its directness. A better example would be a library in which the titles of some of the books were partly obscured. The reason for this was illustrated by the elusive edge operator. The term “operator” as it is usually used in computational vision denotes, in mathematical terms, a linear operator with local spatial support. Although the design of nonlinear operators was also attempted (62), their success was little better. In general, it is impossible to design an operator that responds iff a particular feature is present. Rather, they respond partly to whether a feature is present (the mathematical problem here is related to noninvertibility of operators and  $L_2$  matching theory). Somehow these responses must be interpreted, and it is in these interpretation processes that the general-purpose constraints are embodied. It has been discovered that much of this interpretation can be carried out in mechanisms suitable for parallel implementation, thereby dealing with spatial complexity in a distributed fashion. Before doing so, however, it is essential to stress the change in viewpoint from segmentation to description.

**From Segmentation to Description.** The corners and occluding boundaries that arise in the blocks world are only a small subset of the diversity of physical events of interest in the natural world. Some of these are primitive events, like the change in orientation at the corner of a cube, and others are compound, like the texture of a forest. Thus, it becomes essential to bring out the many levels of structure if there is to be any hope of eventually agglomerating them. The more explicit they are, the easier (in a computational sense) they will be to use. Borrowing insights from another area that was attempting to grapple with complexity—structured programming—Marr (63) proposed three principles to underlie the development of vision systems. The first of these was a principle of least commitment, or the postponing of limiting decisions as long as possible. This is another way to pose the blocks-world criticism. The second principle was one of explicit naming, in which the distinct entities of importance to vision are named so that their description is easily referenced. It, too, deals with complexity. Finally, there is the principle of modularity, the reasons for which were discussed above in the context of organization. In early vision modularity is realized most often by decomposing operations in space as well as into levels.

**Rise of Parallelism.** In the discussion of lateral inhibition it was pointed out that there were two ways to proceed. The first

was toward local (differential or edge) operators, the chosen one. The second direction, or the observation that much of the wetware in early vision appears parallel in structure, is now explored. To understand the advantages of parallel computational machinery, it is helpful to compare it with sequential machinery.

Suppose one wanted to perform a convolution of a particular operator—say, an edge-detection operator—against an image. To obtain numbers, suppose further that it takes  $\eta$  operations to evaluate the convolution at each location. Specifically, suppose the operator consists of a mask whose spatial support is  $n \times n$ , or  $n^2$ , locations. Then, by the convolution formula, there are  $\eta = n^2$  (multiplications at each point) +  $n^2$  (additions) to add them up. Then, for a  $100 \times 100$  image it would take  $10^4\eta$  operations to perform the entire convolution of the operator against the image. For biological “machinery” this is quite a long time. But rather than spend this amount of time performing all of the individual convolutions one after another, biology seems to have evolved a faster solution: Perform all of them in parallel. This trades the cost in time for one in space (special-purpose hardware) but provides the answer in the time required for only  $\eta$  operations. In almost all situations this is very fast indeed.

Researchers in computational vision were anxious to understand and, if possible, to capitalize on the parallelism constraint. There were two problems to face: first, what classes of algorithms could be decomposed into parallel ones and, second, how could knowledge be imbedded (represented) within them? It is rather straightforward to show that parallel convolution machines can be built: Take a large number of processors, or “units,” and arrange them so that the units are connected to their neighbors. Then weight the interaction connections with the coefficients that define the convolution operator and simply have the units perform the required multiplications and additions. Hierarchies or layered collections of such units could then be conceived, with interconnections between units established both across and between layers. This is the standard view of parallelism, and commercial hardware to accomplish such convolutions is now widely available (64).

**Neural Modeling.** But the promise of parallel networks is much more than just measurements and convolutions. An abstract characterization of neurons into binary  $\{0, 1\}$  units led Pitts and McCulloch (65) to discover relationships between neural networks and a particular logical calculus. Other early researchers attempted to introduce some of the uncertainty apparent in neurons (66) and to deal with fault tolerance (67,68). Perceptrons (qv), or linear-threshold devices were introduced in the 1950s as devices actually capable of substantial pattern recognition, and Hebb (69) suggested, in an exciting book, how neural assemblies could underlie much of behavior and learning. But, unfortunately, most of these earlier claims about perceptrons were based on technical notions that have turned out to be inadequate (49), and although much of the earlier enthusiasm remains, new conceptual approaches became mandatory.

**Cooperative Processes and Energy Minimization.** Another conceptual approach to parallel processing emerged from two sources, one in psychology and the other in AI. The psychological contribution is considered first.

Stereopsis (see Stereo vision) is the process by which information about the depth of objects in the 3-D scene is extracted from relationships between the two retinal images. By trigonometry, a point in depth will project to different positions on

each retina, and this difference in retinal disparity is proportional to depth. Thus, depth information could be inferred if retinal disparities could be computed, but this requires the establishment of correspondence relations between structures in one image and those in the other that derive from the same physical event.

Two insights into what parallel processing could do were provided by two (seemingly) different approaches to the stereo-correspondence problem. The first of these was obtained by Julesz (70), who envisioned the entities in each eye as different kinds of abstract "magnetic dipoles." This then implied that the dipoles could cooperate with one another as they relaxed into an equilibrium configuration. This is, of course, analogous to the Ising model of ferromagnetism (71) in which the local-interaction terms are given by the equations of magnetism restricted to nearest neighbors. Such magnetic-interaction terms model an abstract "affinity" or "compatibility" between local pieces of the retinal image.

The second approach is based on another metaphor from physics. Consider a mountain of sand and a billiard ball, and think of the mountain as representing an "energy landscape." If the ball were rolled down the mountain, intuitively it would seek a minimal-energy position (72). The Gestalt psychologists observed early on that such notions could be applied to model vision. Consider the way that Sperling (73) first applied minimization to the stereo problem. He derived his "energy landscape" from considerations about similarity between retinal images. Correspondence can then be viewed as a process of finding the disparity matches (or correspondences) between images that maximizes a measure of their total similarity.

Since stereo correspondence can be formulated in both ways, it is reasonable to conclude that they are two ways of expressing the same thing. Although it may not be clear what the exact relationship between Julesz's and Sperling's approaches is without writing the equations in full, one can see that Julesz's approach concentrated on local interactions whereas Sperling's approach concentrated on their global "sum." (Some literary freedom is taken here, since Sperling also reduces his model to an interactive network. This paper is especially interesting to read for the early view of connectionism that it proposes.) But an essential ingredient was still missing: How could optimization problems of the sort in which Sperling was interested be solved by the kind of local interactions in which Julesz was interested? The search for the answer goes back to the emergence of parallelism within computer vision. Fischler and Elschlager (74) indicated that the direction should be toward abstract structural matching.

**Constraint Satisfaction and Discrete Relaxation Labeling.** To return to computational vision, allow the pendulum to swing from technique back to task. Building on the work of Guzman (75), Clowes (48), and Huffman (76), consider an observation made by Waltz (77) about using knowledge in vision systems. Waltz and the others above were interested in the high-level side of a vision system designed to function in the blocks world or in aspects of what might be thought of loosely as the high-level side of Roberts's system. They were concerned, in particular, with what is called line labeling, or assigning semantically meaningful labels to the lines in a line drawing. Consider, for a moment, how such lines could arise physically. The outside edge of a cube occludes the background, and the edges between visible faces represent surface orientation changes. It would seem, then, that at least in the blocks world,

it is possible to enumerate all of the ways in which physical edges (or their line representations) could arise. Waltz's task was to assign such representations to the lines in a line drawing so that these could then be synthesized into objects and their interrelationships. The synthesis was to be accomplished by search (qv): try all combinations, say in a depth-first (see Search, depth, first) manner, until a match is found. Unfortunately, the simplicity of the search is overwhelmed by the combinatorics.

Necessity therefore motivated Waltz to shift his attention from the global task (which was now precisely formulated) to its local constituents. Waltz observed in particular that much of the search was unnecessary; it went into examining combinations that were in principle impossible. In fact, many of these impossible combinations could be detected locally, and then pruned, before the full graph was searched. All that was needed were rules, say, about how lines could combine at junctions, to detect many physically impossible situations. Thus, in order to complete his search, Waltz implemented a sequential process that wandered around the graph of combinatorial possibilities, deleting impossible ones. The efficiency was thereby improved to the point that the global searches could be completed after all locally impossible combinations were removed.

The next step was to show that the sequential elimination of labels could be done in parallel (78). This step involved a shift in concentration from the task—labeling line drawings—to the technique. It yielded an algorithm in which, intuitively speaking, each label looked around at all of its neighbors and determined whether it was compatible with each of them; i.e., whether there was (at least) one interpretation (of the possibly many) that could be associated with each line meeting at a junction such that the pair formed a local combination that was realizable in a physical object. If not, the (inconsistent) label was discarded. Of course, this inconsistent label may have been the only one supporting another label on one of its neighboring lines, so that the check for local consistency had to be iterated. In this way information about inconsistencies propagates throughout the entire graph. Such a process was first known as discrete relaxation and has now developed into the study of constraint satisfaction (qv) (see also Waltz filtering).

**Continuous Relaxation Labeling.** The above algorithms are symbolic in the sense that they deal with explicit symbols (say, occluding edge) associated with explicit image structures (say, a line). The image structures are easily abstracted to a graph, and the problem then becomes one of labeling a graph; or, more precisely, selection from among a set of labels (symbols) associated with nodes in the graph of a particular subset of labels that is consistent according to relations defined over pairs (or triples, etc.) of labels associated with neighboring nodes. The selection need not be done solely on the basis of discrete relations, however; and the labels need not simply be an unordered set. Rather, continuous measures can be distributed over the label sets at each node, and the label-to-label relationships can be continuous rather than all-or-none functions. This essentially establishes a connection back to the subsection above on optimization and replaces the idea of discrete updating with a continuous, analog-type process. That is, the selection can now be done by maximizing a global criterion function of appropriate form or by solving a variational problem with a particular structure (79). Such processes have been called relaxation-labeling processes, in analogy with relaxation techniques for solving systems of differential equations,

and their analysis and application in early vision has been widespread (80,81).

**Tasks, Tools, and Techniques.** At this point in the discussion it is worthwhile to clarify a distinction that is often confused in computational vision: that between tasks—e.g. determining what is an edge—and techniques—how can edges be detected. Marr (1) put it slightly differently: He claimed that one needed to make a distinction between the problem and the algorithm for solving the problem. He further distinguished between the algorithm and the implementation of the algorithm. The discussion in the preceding subsections evolved into one of tools and techniques and can be summarized by general queries of the form: What class of computations can be implemented on parallel, distributed hardware? Such investigations should lead to algorithms and their analysis. Two distinct kinds of analysis are, in fact, necessary. The first kind relates classes of algorithms to abstract characterizations of techniques; for example, there are many algorithms for solving linear-programming problems or for solving optimization problems. The second kind of analysis relates to properties of a particular algorithm (or class of algorithms): will they converge; are they sequential or parallel; are they numerically stable, etc.

Although there is more on Gestalt psychologists later in this entry, it is worth noting at this point that it can be devastating to jump to particular conclusions regarding how algorithms can be implemented. The Gestalt psychologists took the electromagnetic metaphor quite literally, and when electrical potentials were discovered in the brain, they assumed that their minimization metaphor had been substantiated biologically. They thus took it quite literally for many aspects of brain and behavioral function, and the movement suffered substantially as a result (82).

Although Marr advocated treating problem, algorithm, and implementation separately, there are clearly important relationships between them. The study of tasks interfaces with the study of algorithms through the abstract characterization of what they can do. For example, if edge detection could be formulated cleanly as an optimization problem, appropriate optimization algorithms could be chosen. However, if it is not formulated cleanly and does not work, it is impossible to uniquely ascribe blame to either the task specification or to the technique proposed to solve it. Often both are at fault.

In general, it has been the case that there is a sort of pendulum of activity swinging between a concentration on tasks and on tools and techniques. Of course, to actually accomplish anything, one must pay attention to both issues: what is the task and what techniques can be applied to solve it. It is just this duality that keeps the pendulum swinging. Whenever a particular approach fails, either the formulation or the fabrication (the task or the technique) must be blamed, so the researcher then swings over to the other side. Of course, leaving out highly engineered situations, it is almost always the case that both the task and the technique have been inadequately formulated, which is why the pendulum keeps swinging! Therefore, to proceed, allow the pendulum to swing back to the task side.

**Vision as "Inverse Optics."** Perhaps because they were physicists, both Mach and Helmholtz considered how images were formed, fields known today as photometry, optics, and physiological optics. This perspective has had an immense influence on shaping the second major paradigm for computa-

tional vision, or what might be viewed as second-generation—or second-paradigm—vision systems. [The first generation, of course, is typified by the Roberts system (22).] The particular—backward, or inverse—way in which photometry entered is illustrated first; then the second paradigm is developed. For consideration of other optical phenomena in computer vision, see Ref. 54.

**Shading from Shape (and Light Source).** Light is emitted from a source, reflects off the surfaces of objects, and, if it is not obscured or absorbed by some intermediate object, is captured by the photoreceptors in our eyes. The standard formulation for matte reflection without highlights is well known to physicists, and Mach used it in his research in the latter part of the nineteenth century. The image intensity  $I$  (at each image point) is given by

$$I = \rho(\mathbf{N} \cdot \mathbf{L})$$

where  $\mathbf{N}$  is the normal vector to the surface at the point,  $\mathbf{L}$  is the light-source vector, and  $\rho$  is a scalar coefficient of surface reflectance.  $I$  is a function of two variables (say,  $x$  and  $y$  retinotopic or image coordinates), whereas  $\mathbf{N}$  and  $\mathbf{L}$  are vectors in 3-D space. Clearly, if one knows the scene (or, more particularly for this special case,  $\mathbf{N}$ ,  $\mathbf{L}$ , and  $\rho$ ), one can calculate the image. In words, the shading in the image of an object will vary in an appropriate way with the object, the viewing conditions, and the lighting conditions. But vision is concerned with running the above calculations backward.

**Shape from Shading.** The inversion of the image-formation process is underdetermined. There are always essentially an infinite number of scenes that could have given rise to a particular image. Somehow 3-D variables must be inferred from 2-D ones. Many different sources of constraint are possible, but those that lead to an estimate of where the light source is (with respect to the surface, say), where the surface is and how it is oriented (with respect to the viewer, say), and what the surface's reflectance properties are would seem to be among the most useful. Recently Horn (83), Woodham (55), Pentland (84,84a), and others have tried to recover information about surface shape from changes in illumination (or shading), developing an activity initiated by Mach. Recently, in fact, an entire industry of approaches known as "shape-from-X," where X can be texture, contour, or motion (85,86), has emerged (see Shape analysis).

Beck (87), among others, has studied such phenomena psychophysically. But since image formation is a complex of processes, it follows that segmentation is not just a matter of image differences but can also be a matter of inferred physical object differences (segmentation is the decomposition of the image into pieces that arise as the projection of distinct physical events whose recovery would enable or support object inferences). Waltz's research was an important case in point; see also Mackworth (52). The next decade (1975–1985) of computational vision research was, to a large extent, an attempt to verify this observation; to calculate and to make explicit these intermediate inferences from images back into the scene domain.

## Second Paradigm: From Segmentation to Surfaces

Image differences can arise from differences in lighting (say, cast shadows), in surface orientation (as at the corner of a cube), in surface composition (as when one object obscures another), etc. The next major focus of computational vision re-



search was exploring these differences both individually and together. Although edge detection seems impossible solely on the basis of image intensities, each of the above individual properties, if it could be computed, could then be differentiated. The result would be a description not only of where intensities changed but also of which (estimated) scene properties were changing as well.

The key difference between this second paradigm and the first one is how the line is drawn between low- and high-level vision. In the first paradigm the goal of low-level processing was a segmentation, or the recovery of a line drawing of (intensity) outlines. Somehow these outlines are to be matched against a database of prototypical object outlines. Such matching is impossible for natural objects, however, since intensity discontinuities do not always correspond with object structure. For example, intraobject texture differences may obscure interobject edge differences. In the second paradigm the attempt is to recover a richer, more abstract intermediate description, namely, surfaces. The difference between the paradigms is that now, in the second paradigm, segmentation is to be carried out with respect to abstract, inferred properties, not only with respect to image-intensity properties. Then the matching can be guided by (abstract) object as well as a priori properties. And this general-purpose, intermediate level of description will incorporate information from many different sources, including stereo, motion, shape from shading, as well as many different kinds of constraints (such as object smoothness).

The influence from psychology is again clear here. Gibson (88), who, like Helmholtz, had very limited ideas about the need for computation early on in the visual process, nevertheless had a clear idea of the importance of inferring abstract "surface" properties from images. To this end he and his students studied motion, texture, stereopsis, and shading or many of the modes through which information about surfaces could be obtained (to be more precise, the statement here is not that earlier researchers, including Helmholtz, e.g., were unaware of these sources of information but rather that Gibson illustrates the enthusiastic revival of interest in them). Each of these appeared to be such a rich source of information to him that one could almost understand why he thought the visual system could resonate to them. It is striking, in fact, to compare illustrations from his highly influential 1950 book and Marr's (1) much more recent attempt to lay out a computational viewpoint. They are remarkably similar! Curiously, the enthusiasm surrounding Gibsonian "resonances" continues to the present; see the commentaries of Ullman (89).

How, then, is it possible to structure the surface-finding processes, i.e., those processes that actually perform the inference of surfaces? The influence from physiology is again strong, as are inputs from mathematics and computation.

**Interacting Modules for Surface Interpolation.** Two approaches emerged within this second paradigm almost simultaneously, one based on a working assumption of independence and the other on one of dependence.

**Primal Sketches.** The first of the frameworks around which second-paradigm computational vision developed was the primal sketch idea of Marr (63). The primal sketch is an explicit representation of the "important information about the two-dimensional image, primarily the intensity changes there and their geometrical organization" (90). The primal sketch is therefore a data structure, and with development it actually

evolved into several data structures: the raw primal sketch, which held the results of "edge detection" (qv); the full primal sketch, in which geometric relationships were first made explicit; and the  $2\frac{1}{2}$ -D sketch, in which certain depth properties were first computed. Model descriptions were then inferred from these data.

Marr took the principle of modularity very seriously and proposed a research program in which each of these different stages, or the processing that actually comprised them, could be studied independently. Separate projects in stereo, motion, and edge detection were therefore begun. Although modularity or independence is arguably just a first approximation, the competing framework stressed interrelationships.

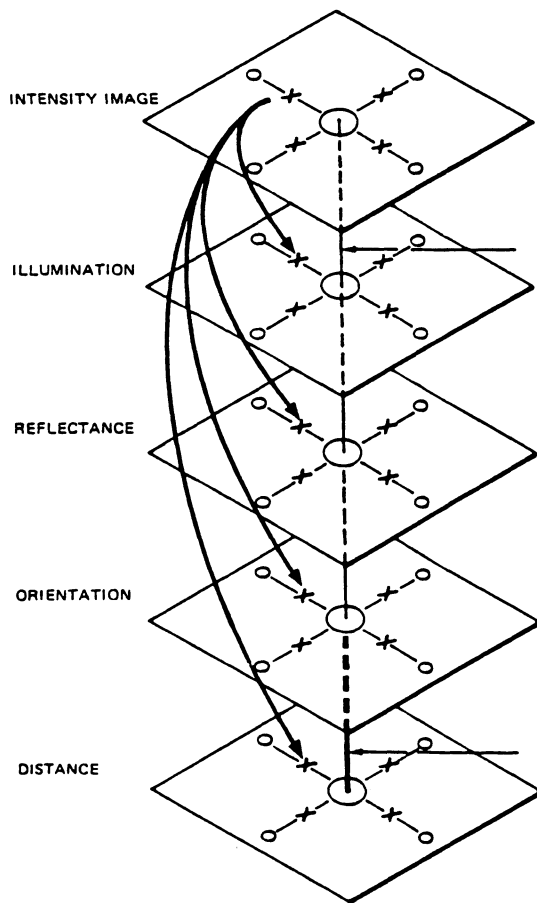
It is interesting to examine the progress from Roberts's line drawings to Marr's primal sketch. Primal sketches are simply an elaboration of the data structure interfaces common to all computer-vision systems. The question was—and is—precisely where they should be placed within the system.

**Intrinsic Images.** The other dominant framework for second-paradigm vision was founded on the idea that since the surface–light source–viewer arrangements were all intrinsically coded within images, the role of early vision was to explicate them. Hence Barrow and Tenenbaum (91) proposed the idea of intrinsic images, or a collection of arrays aligned in retinotopic (image) coordinates. Each array made one of the local intrinsic properties explicit, including image intensity, surface normal, distance (or disparity), and lighting. Since none of these intrinsic images could be computed by itself, functions were further defined between them which embodied, say, the photometric relationships that must hold between them. Each image further made discontinuities explicit, so that this information could be propagated between images as well (see Fig. 9).

**Slicing Up The World For Constraints.** The approaches to vision so far have indicated several different ways in which assumptions about the world can be obtained. First, it is possible to have a complete but artificial world. This is the case for the blocks world, from which we can conclude that although some quantitative analyses thereby become possible (say, Horn's image-intensity calculations), the results do not extend to more general universes. A second way to slice up the world is by introducing intermediate constraints that hold for some aspects of the general world. Examples of this are smoothness of surfaces (92,93), the continuity of edges (94,95) and the rigidity of objects (96–98). But beware, when selecting assumptions, that they do not conflict with the quantitative requirement above: There is almost a sensitivity principle operating here in which the consequences of violating an assumption should vary in proportion to the universe over which it holds.

**Symbolic Side of Early Vision.** In addition to making intrinsic properties explicit, the above framework began to confront the symbolic content of early vision as well. Implicit in the previous generation was an assumption that each stage was interpreted—in the logical sense of the term—by a subsequent one. Now researchers began to worry about the semantics of these symbols; i.e., about what they meant (99,100). Early vision was shown to involve inferences about "what's out there." Such concerns developed from the concern in high-level vision with symbols and appeared explicitly in mecha-





**Figure 9.** Typical second-paradigm computational model, in this case Barrow and Tenenbaum's (91) intrinsic images. The planes are intended to represent various intrinsic features, such as surface reflectance, arranged retinotopically, and the arrows indicate quantitative constraints between them (equations connecting them). But such purely quantitative, global models now appear to suffer from intractable stability and conditioning problems.

nisms such as relaxation labeling. There is an essential difference between relaxation labeling as it emerged in the middle 1970s and neural modeling, which is much older; in relaxation labeling the labels are symbols. The importance of meaningful symbols, and the inferential processes that manipulated them, increased from this time on in all of vision.

**Realization of Second Paradigm.** The realization of the second paradigm has been attempted for the past decade, and hence much of the research is covered in detail in more specialized chapters in this *Encyclopedia*; see, in particular, Color vision; Dot-pattern analysis; Motion analysis; Scale-space methods; Template matching; Edge detection; Generalized cylinder representation; Image analysis; Optical flow; Scene analysis; Shape analysis; Stereo vision; Texture analysis; Visual depth map.

**Organization of Primate Visual Systems.** Decomposition of function arose in physiology as well as in computer vision. One certainly has the feeling that such anatomically apparent modularity lies behind the strong form of modularity advocated by Marr (1). Consider the macroscopic organization of the primate visual system. The first level of organization is rather coarse and can be characterized in terms of anatomically dis-

tinct areas to which the visual input is mapped. Recognizing that perhaps a few new visual areas will still be discovered, there are something in the range of 15–20 that have been at least partly characterized (101). It is tempting to relate a visual function to each of these areas, say, with one solving stereo, another solving edge detection, etc. It is also the case that different neural layers within each of these (anatomical) areas may be involved in different functions, which adds up to another factor of 3 or 4. However, this relating is dangerous since each visual area may well be involved in more than one function; recall, e.g., that simple cells are both orientation- and velocity-tuned. An even more telling example has been found by Regan and Cynader (102) in cells whose receptive fields are both motion- and disparity-sensitive. That is, receptive fields exist whose stimulus dependencies involve both velocity and disparity simultaneously; they cannot be decomposed. Even keeping such compositions in mind, however, a first-order estimate of functional complexity is possible. The number of distinct visual areas is significantly larger than 2 (for separating low- and high-level vision) and less than, say, 100, even taking the different layers into account.

The detailed organization of each of these visual areas exhibits structure of its own. The simple, complex, and hypercomplex cells of Hubel and Wiesel (33) are found in the first few cortical visual areas. Fascinatingly, at least in the first of these visual areas (V1 in primates, area 17 in cats) these cells are not arranged randomly but rather form striking columnar patterns according to a number of criteria. First, there are ocular-dominance columns in which cells take their dominant input from the left or the right visual field. Then there are orientation columns in which cells have a preferred orientation that changes sequentially along tangential (electrode) penetration. Finally, the receptive fields vary in size over several orders of magnitude, with some less than  $10'$  (minutes of visual angle) and others more than  $10^\circ$ . In general, complex cells have larger receptive fields than simple cells. It would definitely seem that this organization provides strong support for parallel processing across spatial (and other) dimensions, although the mapping from anatomical area to visual function is still very obscure.

**Parallel functional streams.** Although only a few of the properties of cortical receptive fields have been considered, the story is undoubtedly more complex than this. Returning to the retina, the original view that ganglion cells only exhibit spatial circular-surround receptive fields has been replaced with a more modern one in which their temporal characteristics matter as well. Even in the classical view there is evidence of two parallel-processing streams with respect to contrast: one which is ON center/OFF surround, and the other which is OFF center/ON surround. It is now widely believed that there are many different classes of retinal ganglion cells, X and Y being among the most prominent. X cells are roughly linear in their response and have relatively slow connections back to cortex. Y cells, on the other hand, are nonlinear in their response, have fast connections, and have receptive fields roughly twice the size of X cells. Both scale in size with retinal eccentricity. Also, there is a third class of so-called W cells that is beyond the scope of this entry.

Although functional ideas regarding the X and Y systems are still hypothetical, it is held that the X system appears to be more concerned with spatial vision, whereas the Y system may be involved in motion (103). However, the significant point is that the X and Y pathways proceed in separate, parallel chan-

nels from the retina through the LGN to the cortex. There is even evidence that they give rise to separate populations of simple and complex cells as well. Thus, in addition to the hierarchical, layered organization (see Naive Physiology (above)), visual processing would appear to be accomplishing several functions in parallel as well. Furthermore, physiological evidence points to separate enervation to (at least some) simple and complex cells, which brings the strict hierarchy into question. And more recently the processing of color has been relegated to a (partially) separate system (104). The feeding forward and the feeding back of visual information does not follow the simple branches of a tree but rather flows around a graph. Complexity rises again.

What kinds of functions could the neurons along all of these parallel, layered pathways be computing? Although the separation of color and luminance information makes evolutionary sense, and the separation of contrast makes mathematical sense, how can other visual functions be decomposed into parallel streams? It is fascinating that this is precisely the question that began to dominate researchers in computational vision, for different reasons, during this second paradigm. Decomposition and organization are the only ways to deal with complexity.

**Summary of the Second Paradigm.** In summary of this section, the second major paradigm in computational vision, it is interesting to stress that the partition of early and later processing, which began with Helmholtz (19) and Roberts (22), has continued. The primal sketch and intrinsic images are far more complex interfaces than Roberts's line drawing but nevertheless serve the same putative function. They certainly contain much more information, which underlines the complexity of early vision, a point that has emerged many times. But the metaphor of photometry on which this style of analysis is based has become questionable. Just how far quantitative methods can be pushed remains an open question. The missing ingredient is qualitative structure.

### General-Purpose Models Revisited

Although the modular view has many attractions (recall the argument above regarding the Dewey decimal classification), the formulation it assumed within the second paradigm has serious problems. A decade of concentrated effort has not led to generally functional systems; rather, there have been only "local" successes, or successes only under many explicit and highly restrictive assumptions. The situation is reminiscent of the blocks-world experience, in which knowledge of (or assumptions about) the physical situation were so restrictive as to be suffocating.

The problem with second-paradigm vision systems can be seen from the discussion of "inverse optics"; for the program to succeed, all of the details of surface normals, surface-reflectance functions, spatial arrangements, etc., need to be recovered exactly. There is no place for approximation, noise, or uncertainty. But approximation, noise, and uncertainty are present in any real physical vision system. The strong form of the second paradigm cannot succeed.

The examples from perception presented above also argue strongly against the second paradigm. The Mueller-Lyer illusion (Fig. 1) and Mach bands (Fig. 2) indicated that what is seen is not exactly what is there; rather, it is the result of a

context-dependent computation. And the subjective Kanizsa edge (50) (Fig. 7) suggests that the computation involves a form of inference.

If the exact structure of the scene cannot be recovered everywhere, which parts can be? Which aspects of structure are necessary (sufficient?) for visual inferences? Thus, the question of what kinds of knowledge are employed, and to what ends, rises again. There are two major issues in particular: whether the knowledge is truly quantitative and how to slice up the world so that appropriate general assumptions can be made.

**Qualitative versus Quantitative Knowledge.** The fact that image-intensity formation is a complex process led to an early observation that intensity profiles are not always shaped like a step function but also arise as roofs or slopes (28). The structure of these intensity functions carries important information about the physical scene that gave rise to it (53), an observation that was instrumental, in large part, for the concentrated effort on "inverse optics." But much of this research was quantitative in motivation (if not in fact), with the goal of precisely formulating the systems of differential equations and solving them exactly. Unfortunately, as Mach indicated, this is impossible in general, and those restrictive situations in which it is may not be all that relevant to the solution of the general vision problem. Although photometers must be quantitative in their interpretation of, say, shading profiles on the moon as seen through telescopes, they are functioning as physicists and their goal is to recover the surface topography of the moon as accurately as possible from that source of information. It would appear that vision systems cannot accomplish this in general for all scenes.

**Qualitative Shape from Shading.** Because of its central position, the methodological presupposition within the second paradigm that one's visual system functions as an inverse photometer must be questioned. The evidence, in fact, is against it from a biological perspective. For example, if computer-graphics techniques were used to render images of cylinders with different cross-sectional profiles, ranging, say, from circular to triangular, one would expect the percept of the figure to change as well. This does not happen, however (105); rather, a range of different surface shapes are all perceived as roughly circular. The impression is that shading cues are more qualitative than quantitative. Cavanagh (106), in a recent psychophysical study, has shown that depth cues such as occlusion and shading interact only in certain ways, depending on whether they derive from color, motion, stereo, etc.

The next example raises another kind of problem with strict quantitative constraints in early vision.

**Rigidity of Rigidity Assumption in Structure from Motion.** Humans have the remarkable ability to see structure from moving-dot patterns (96,97). There is, of course, much more to motion than is considered in this entry; consult the other articles in this *Encyclopedia* for a more complete treatment. Imagine, e.g., a clear cylinder covered with tiny specks of dust. Statically, the dust pattern would appear to just be random dots arranged on a plane. If the cylinder were rotated, however, its full 3-D structure would be apparent. Ullman (98) has referred to this ability as inferring structure from motion, and a good deal of research, both computational and psychophysical, has been focused on it (96,97,107, and 108). In particular, the apparent perceptual preference to see rigid configurations

has inspired researchers to build the rigidity constraint directly into the computation, leading to models in which systems of algebraic equations need to be solved exactly.

However, the computations arising from such formulations are difficult, if not impossible, to realize biologically or perceptually because of a numerical problem called ill-conditioning. A system of equations is ill-conditioned if a small change in input (independent variable) leads to an arbitrary change in output (dependent variable). Such small changes in input arise from errors in measurement or lack of high numerical precision, and hence such quantitatively sensitive algorithms seem ideally unsuited for real vision. Visual systems must be designed to function in the presence of noise and uncertainty. Global rigidity assumptions are too rigid; they fracture under the slightest pressure from uncertainty.

**General Algorithms and General Position.** The above point about rigidity can be made in different terms by the notion of general position. Imagine viewing a line drawing of a cube. From all positions except one this line drawing will be topologically similar; there is one, however, at which it looks different: when the corner nearest the viewer and the corner farthest from the viewer align along the viewing axis. In this singular configuration that line drawing no longer looks 3-D; rather, it looks like a flat triangulated hexagon (see Fig. 10). Note, however, that such singular configurations are destroyed if one moves ever so slightly off the axis connecting the two corners. In all but this one configuration the cube is viewed from a general position. By definition, then, a general-positional view of an object is one in which the image does not change qualitatively. Other examples of singular arrangements could be obtained if one were to view scenes from singular viewpoints; recall the Ames room demonstrations (109). They did not look the same from any other viewpoint. Vision algorithms must be general in precisely this sense; the rigidity assumption is analogous to a singular view.

#### From Structure to Function

In the beginning of the entry, two pendula are described, one of which indicates the tension between low and high-level knowledge and the other between task and technique. Although the field has become much more experienced, the pendula still remain. Ten years ago there was great concern about whether processing was top-down or bottom-up; now it is what kinds of knowledge must be applied; what are legitimate assumptions to make and what constraints do they imply. However, the assumptions and constraints are usually ones of principle; e.g., the assumption of rigidity in motion (97,98). How



**Figure 10.** Two line drawings of cubes. The one on the left illustrates a cube from an arbitrary viewpoint; although the details change, the topological features remain the same. On the right is an illustration from a singular viewpoint, in which the three-dimensionality may or may not be present. Any slight change in viewpoint destroys the triangulated hexagon and results in a drawing like the one on the left.

should they be made discrete in practice? The need for modularity has arisen within both tensions; the question has become which structures in the scene project into which structures in the image (informational modularity), and how might these be recovered reliably and consistently (processing modularity)? Spatial parallelism is clearly indicated in the early stages; how high it goes is still an open question (110,111).

The idea here is more than that certain scene structures are preserved under projection. Helmholtz (19) and Gibson (88) were both aware of this (the classic examples from Helmholtz concern how straight lines in space project into straight lines in the image; Gibson, of course, discovered texture gradients and optical flow). The idea is that vision consists of a fabric of inferences, some or many of which are interrelated. Not all of the scene information is recoverable directly; rather, only some kinds of structure in the scene domain give rise to image structures from which the "projection" transformation can be "inverted." It is this network of local (in every sense of the term) inferences on which the global scene recovery is anchored. Somehow the global structure must be interpolated from the local anchors. It is the discovery of such structures that provides the keys into visual function.

**On the Mechanisms For Early Inferences.** Three major lessons have been learned about how to approach early vision: the mechanisms for structuring the early inferences must incorporate assumptions that hold abstractly over significant subclasses of real-world structure; they must be as insensitive to noise and uncertainty as possible; and they must be utilizable within distributed "hardware" (either networks of neurons or VLSI circuits) to deal with the complexity of size. It has been slowly realized that such inferences can be carried out by distributed optimization, relaxation, or related machinery. The constraints are more satisfiable, when posed within minimization- or hypothesis-testing frameworks than as rigid systems of equations (112–115). And since constraints on these inferences exist from many different sources, the mechanisms within which they are utilized must permit their interaction as well.

**Local Structure of Intensity Discontinuities.** Both issues—how qualitative and how encompassing assumptions are—emerge in the search for the elusive edge-detection process (recall The Elusive Edge Operator). Concentrate on the first lesson listed above: that there should be a significant but "local" problem, i.e., a significant amount of knowledge about the local structure of images that provides a solid foundation for inferring a local piece of the scene that gave rise to it. Since the range of physical events is large that can project into what one should like to call edges in the image, it follows that the description of the edge must be rich enough to support decisions about the physical cause (edge events can arise from surface-reflectance changes, surface-orientation changes, occlusions, and lighting changes taken individually or in combination). Simple image differentials are unlikely to work, since they only localize changes in intensity (e.g., Ref. 116). More complex processing is needed for determining not only the locations of the image-intensity discontinuities but also the shape of the intensity function in a local neighborhood around it.

The importance of the local structure of image intensities in the neighborhood of an edge has been realized since the blocks

world. Binford and Horn (29) matched intensity profiles along an edge; however, their matching relied heavily on assumptions about the blocks world (e.g., planarity of surfaces). More recently Quam (117) used the intensity structure across linear features to track roads, and Witkin (118) correlated intensity structure across putative edges to verify their presence. Most important, however, Witkin tried to use these across-edge correlations to infer something about the physical event that gave rise to the edge, observing that when objects stand in an occlusion relationship, the pattern of intensities on either side of the occlusion should be similar, but not necessarily across the occlusion. Such information clearly supports the mapping from the image back to the scene under certain circumstances.

One can go further. Not only should intensity structures be similar on either side of an edge, but their detailed structure can be used to support inferences about the edge (41). To illustrate, recall the above discussion of shape from shading. Consider a matte surface smoothly varying in orientation. By the image-irradiance equation it follows that the projected intensity function will be smooth as well. Now, suppose that a shadow were cast on the surface. This would add a step to the (log) intensity distribution, resulting in an image structure in which the slope of the intensity function would be the same (and almost certainly not flat!) on either side of the discontinuity. Therefore, to locate this type of edge, and to distinguish it from others, it is necessary to describe both the intensity function and its slope on either side of a discontinuity. Other schemes that simply locate discontinuities at the cost of irretrievably modifying the local intensities cannot support inferences back to the world.

Leclerc and Zucker (41) have developed a nonlinear scheme for accomplishing this estimation of local intensity structure concurrently with detection of discontinuities. It involves spatial interactions between convolutions of similar sizes as well as "level" or scale interactions between different sizes. Such intra- and interscale interactions are necessary because of the trade-off between neighborhood size and noise immunity: the larger the neighborhood, the better the performance at detecting and describing an edge. But the performance will degrade if more local edge events are smoothed over in the process. The network interactions are necessary to guarantee that this does not take place.

Edge detection is therefore not just a matter of finding the right edge operator but rather requires understanding the interactions between measurements as well. This leads to inferences. The structure across edges has been stressed—the local structure of intensities in the neighborhood of discontinuities; now a different aspect of the structure is considered—that along curves rather than across edges. This is dealt with by differential geometry.

**Analysis of Orientation: Curve and flow recovery.** As another example, consider the (related) problem of curve detection. Curves are an important subclass of structure in the world because they satisfy the above criteria: They occur in almost all images (as occluding contours, surface creases, hair and other surface coverings, etc.), and they provide information on which subsequent surface inferences can be based (115, 119). How can they be recovered, or inferred, from images?

Early approaches to curve detection were barely distinguishable from early edge detection. Simple cell operators were convolved against images and maximal values selected by thresholding. This is because they respond maximally

when centered exactly on a line and oriented similarly to it; hence they have been called line detectors. Unfortunately, they run into the same problems as "edge" detectors.

**Lines versus Tangents.** What relationship do so-called line detectors have to the actual detection of a curve? A logical place to start is by asking what a curve really is: Mathematically, a curve is a function that maps an interval of the line  $\mathbb{R}^1$  into an embedding space (say, the plane  $\mathbb{R}^2$  or space  $\mathbb{R}^3$ ). What is given in an image is not a curve, but rather a discrete sampling of the trace of a curve, or a set of points through which it passes. Curve detection is the process of inferring a curve from its trace subject to additional constraints.

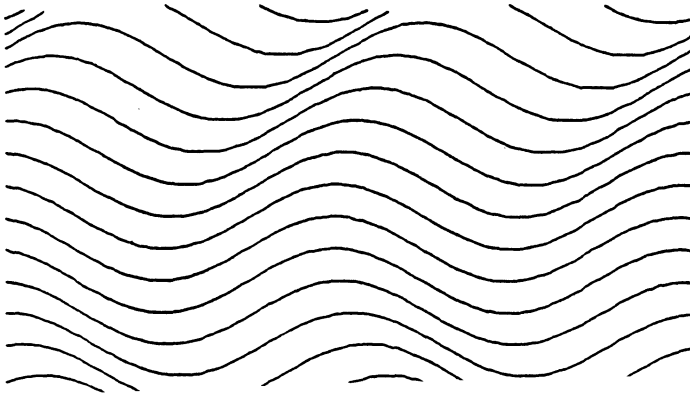
It is important to view curves abstractly and mathematically rather than pragmatically as image structures because it suggests that the intermediate structures should be abstract as well; in this case lines are not what need be detected but rather tangents to curves. Curvature can then also be shown to play a role, so that tangent fields can be recovered by minimizing a functional of curvature variation (115). Since the tangent is the first derivative of the curve with respect to arc length, the global curve can be easily recovered from such local representations of it by a process of integration. Tangent fields are an arrangement of discretized and quantized tangents in retinotopic coordinates.

If the problem were viewed as one of line detection rather than tangent detection, subsequent stages (smooth global curve inferences, placement of corners and discontinuities, etc.) would be difficult to formulate in a mathematically consistent manner.

Viewing curves abstractly also raises additional possibilities for interpreting scale as well. Recall the traditional view that larger operators detect larger events (recall Edge Detection and Scale). However, since curvature is a relationship over neighboring tangents along a curve, it suggests that perhaps the role of the larger operators is related to these higher order properties (115). One should certainly expect this to be the case mathematically, and computationally it works as well. Indications are that the biological equivalent to curvature measurements are embodied in the hypercomplex cells (see Naive Physiology, above) or, more precisely, in their "end-stopping" property. This is a case in which the explanation of biological data together with basic mathematics has yielded a successful computational-vision algorithm (115); but see also Ref. 120.

**Parallel Surface Contours.** Consider a surface covered in parallel pinstripes. These will project into "parallel" curves in the image. Stevens (119) studied the inverse situation, or the inference of a surface from a collection of "parallel" curves. Such displays give strong impressions of surfaces when viewed (see Fig. 11).

**Flow Patterns.** It is rarely the case that the surfaces of objects in natural scenes are covered with regularly arranged contours, however. The more natural case is that the curves are arranged so densely that they cover the surface in a physical sense, often winding in and out of occlusion relationships. This is the case for hair and fur patterns consisting of only roughly parallel arrangements of curves (hairs), or in the case of motion, such flow patterns arise in waterfalls, or when the projected image of a complex physical arrangement changes rapidly, as when one runs through a forest. Note that hairs are different from the pinstripe patterns discussed above in that pinstripes almost never touch and are continued for a long distance, whereas hairs almost always touch and are rarely

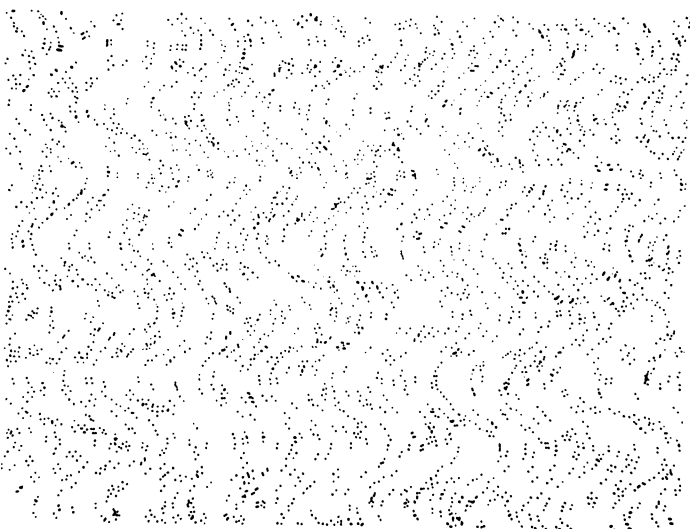


**Figure 11.** Illustration of how parallel surface contours (pinstripes) provide a strong cue for surface orientation.

visible for long distances. Somehow the surface covering (the hair pattern) must be recovered from the (relatively) sparse information available in the image as highlights.

The issue here is another abstract mathematical one, and the way in which it connects image events (patterns of intensities) to events in the world (patterns of hairs). Topologically, curves are 1-D constructs; surface coverings are, like surfaces, 2-D constructs. One should therefore expect the recovery of surface-covering descriptions to require processing in addition to that required for curve inferencing. In particular, the recovery of orientation information for flow patterns involves a direct form of interpolation, or the spreading of curve information, to "fill in" the areas between "highlights" of projected image structure (see Fig. 12). The result is a tangent field, or description of the orientation information, over an entire 2-D.

**Texture.** There are other regularities in addition to those in image structure that are connected to orientation. Such ar-



**Figure 12.** Flow pattern covering a surface with the same kind of undulations as the one in Figure 11. Now, however, the covering is more like a hair covering. Local indication of orientation is provided by pairs of random dots; global 2-D organization is inferred by one's visual system. Note that, unlike Figure 11, there are no long contours and that they are not evenly spaced but rather appear to cover the surface densely.

rangements are called textures (88), and their structural geometric regularities have been modeled by Zucker (121). Beck (122) has shown that most of the perceptually relevant structure of (static) textures is contained in intensity, orientation, and color distributions, and Haralick (123) has reviewed the different approaches to computing texture descriptions.

If textures are that class of image structures that indicates regularity, at least in some sense, the other end of the extreme are those image structures that indicate (and arise from) chaos; consider, e.g., clouds, trees covered with leaves, and the surf. Pentland (84) has studied how such fractals can arise in images and how their structure can be inferred.

**Surfaces.** Once information about surface coverings, contours, and edges is available, the question arises regarding how to infer surfaces (or other abstract structures) from them. Many other examples of such inferences abound; recall the discussion of shape-from-X methods above (Tasks, Tools, and Techniques).

To illustrate the technique, recall the observation of Helmholtz that straight lines in space project to straight lines in images. Now, if it were known that the lines in space were regularly arranged, say as a square grid or as (in the case of contours) circles, their projection onto a surface would give rise to a regular geometric distortion; the square grid would go into a trapezoidal one, and the circles would go into ellipses. Hence, given these figures, the projection equation could be inverted for them (85,124). Again note that this is not a case of doing inverse optics in general but rather doing it for a local problem (a specific grid). Such ideas underlie much of the 3-D vision in robotics (qv), in which structured light arrays are projected onto objects using either patterned sources or lasers (125).

Kass and Witkin (126) present a more radical attempt at structural inference; they would like to characterize the process by which objects are formed, as, e.g., the flow of tree bark around a knot.

**Corners and Parts.** At a level of abstraction up from curves and edges, even sparsely supported, are structures that are derivable from them, say their discontinuities, inflexions, and extrema. For a discussion of how to detect corners, see Brady and Asada (120), Davis (58), and Zucker (115). In a collection of parallel contours, e.g., corners that align along smooth contours provide a curve inferencing problem one level higher (than the curve inferencing), and similarly these corner contours could be grouped. The study of such grouping processes is related to the way in which complex objects are decomposed into parts, an area currently under investigation (95,127). Intersections between objects almost always map to singularities in the bounding contour, for example.

**Figure/Ground and Structural Grouping.** The Gestalt psychologists emphasized the role of structure and organization in early vision nearly 65 years ago (e.g., see Ref. 128). They identified a phenomenological level of processing in which figure was separated from ground according to a number of principles, including proximity, good continuation, common fate, etc. But their proposals have defied quantification until now (16). Perhaps the reason is because of the complexity issues that have been discussed here. There are, e.g., many different ways in which curves can arise in the physical world, as is the case for edges, and their subsequent uses are likely different. Bounding contours arise from inferences that link discontinui-

ties with similar local intensity structures, and surface contours may have similar intensities along them. Discussed above is how curves such as surface contours can be detected, but this requires an assumption that the given trace of the curve is dense enough to properly stimulate the initial convolutions. Curves that are sparse with respect to this metric will certainly require different techniques.

Grouping is a term usually taken to denote the agglomeration of local structures into more global, and perhaps more abstract, wholes. The standard example is Wertheimer's (129) observation that collections of points are not seen as points but rather as figures. As discussed above, the Gestalt psychologists believed that the (physical) minimization of real quantities was the mechanism by which this grouping was accomplished, in much the same way that soap bubbles minimize surface area. Although it is now known that minimization can be accomplished by more computational methods (e.g., see the section Relaxation Labeling), the original idea that grouping could be accomplished by a single mechanism is extant [e.g., Marr (1)]. Given the arguments about the diversity of inference mechanisms for curves and edges, however, this seems unlikely, in much the same way that a single-edge mechanism now seems unlikely. Rather, one might guess that there are a diversity of grouping processes, each tailored to different (classes of) function (115). Curve and edge processes are discussed above; a few other examples follow. This argument has now been generalized by Witkin and Tenenbaum (130), Lowe and Binford (131), and Lowe (132). It holds that, in general, structure is unlikely to arise randomly in images; rather the structure of the world is such that, should events somehow "line up," a common cause should be attributed to them. General-purpose models capture this common cause at early, but still abstract, levels.

**Differential Equation Metaphor.** The way in which, say, occluding contours and hair patterns relate to the physical world leads to another difference between them, a difference that can be generalized to include many other classes of structural information. Occluding contours arise when surfaces intersect projectively; they give rise to intersurface constraints. Hair patterns, on the other hand, provide information about the particular surface on which they lie; they give rise to intrasurface constraints. This difference is fundamental to the processes that must put various sources of information together. Again one is led to the metaphor provided by differential equations, in which the solution is governed by two distinct classes of constraints: the differential operator, which constrains how the solution varies over its domain, and the boundary condition, which constrains the domain and the value at the edges of this domain. Note that, for 2-D differential equations, such as Laplace's equation, the differential operator is an infinitesimal function of two variables, whereas the boundary condition is given by, say, the values along a 1-D contour. Infinitesimally such differential operators represent the kind of constraint available from flow patterns, whereas the boundary conditions resemble 1-D contours. In terms of the previous examples, the Laplacian corresponds to the orientation information provided by an infinitesimal "piece" of a waterfall, and the boundary condition corresponds to its bounding contour.

To be more specific, recall that Laplace's equation is

$$\nabla^2 u(x, y) = 0,$$

where  $\nabla^2$  denotes the Laplacian (differential) operator ( $\partial^2/\partial x^2 + \partial^2/\partial y^2$ ). If  $\Omega$  is an open neighborhood, a well-defined problem would be to find  $u$  in  $\Omega$  from prescribed values of  $\nabla^2 u$  in  $\Omega$  and of  $u$  on  $\partial\Omega$ , the boundary of  $\Omega$ . Such problems are known as Dirichlet problems. Within  $\Omega$ ,  $u$  is completely determined by the constraints provided by Laplace's equation and by the value of  $u$  along the boundary  $\partial\Omega$ .

The above example is, of course, metaphorical. This is not to say that intrasurface constraints are Laplacians. Rather, it is the abstract mathematical form that is of concern, and it is not limited to waterfalls. Other sources of static intrasurface constraint come from monocular shape cues, such as shape from shading (150, 151), from binocular stereo disparities (70,93,133), and so on as has been discussed. Intrasurface cues only hold for particular surfaces; they take abrupt jumps as the projected image from one surface undergoes a transition to that from another. Similar arguments hold for the transitions in lighting, which has also been discussed, say from an illuminated area to one in a cast shadow (41). Topologically all of these transitions are 1-D contour boundary conditions that constrain the area over which the other, 2-D intrasurface constraints can be integrated.

**Free Boundary-Value Problems.** Clearly there are many sources of both inter- and intrasurface constraint, including motion, stereo, shading, texture, color, and their differences. I have already argued that their inference will involve interpolation and that "edges" provide the boundary conditions for limiting them. How can these different sources of information be put together? Clearly situations will arise in which the intrasurface information will be ambiguous, as will the intersurface information, and both inferences must occur simultaneously so that they can mutually constrain one another. Intuitively the situation is like computing the shape of a soap bubble over a flexible ring; clearly, the final shape will depend both on the ring and on the soap. Such problems are called free boundary-value problems, and they arise in many areas of mathematical physics (134).

An early attempt to implement these ideas in a simplified vision context involved dot clusters, in which the problem was to label the dots defining the edge of clusters simultaneously with labeling the dots interior to the clusters (135). Separate processes for intracluster and intercluster (edge) labeling were specified, as were their interactions. Briefly, one might speculate that, in the vision context, the intracluster processes would be integrative, region-growing-type algorithms over intensity (concretely) or shape and reflectance (abstractly) constraints; the intercluster processes would delimit the various type of edges between them. More recent attempts at integrating information from different sources in vision systems are discussed by (136). Much remains to be done along these lines.

**Generalization of the Framework.** The framework provided by inter- and intrasurface information, or, in different terms, by differential equations, holds not only for the features described here but also for abstractions over them. Contours arising from abrupt changes, in, say, a flow or hair pattern could provide the boundary constraint to a higher level process. This could correspond to a physical situation in which the underlying surface changed orientation abruptly but the surface markings smoothed it over somewhat. Thus, issues of how to differentiate flows become as important as the flows themselves.



## High Level Vision

The beginning of this entry differentiates between low-level and high level vision by asserting that low level vision is the study of general constraints on special-purpose hardware whereas high level vision is that of special constraints on general-purpose hardware. In the evolution of this entry many different low level constraints are uncovered with nontrivial roots in higher level vision. The earliest work on computational grouping may be Guzman's (75) "matched Ts" for joining pieces of contour occluded by a common block together. As these different constraints were refined and "moved down" in visual systems, the need emerged for intermediate-level structures to support them. Perhaps the earliest tenable idea in this direction is Binford's (137) notion of a generalized cylinder (qv); see also Brady and Asada (120). More specialized constraints are bound to emerge here, in the sense that they are more global. Rather than searching for local structures that can be inverted, here one is searching for a vocabulary of intermediate objects; a kind of modeling language for prototypes. The most recent contribution in this direction is a proposal by Pentland (138) for "superquadrics," an extension of quadric surfaces done for solid modeling in computer graphics (139). But it is not yet clear how these superquadric constructs relate to more traditional graphics modeling primitives (140).

The requirements for intermediate structures are influenced both by what is coming from "below" and what remains to follow from "above." Lowe (132) makes the point that the result of grouping operations should provide indices into model databases, and attention (141), may well provide a connection in the opposite direction. Robotics imposes its own special constraints (e.g., see Ref. 142).

High level vision has a very different structure than what has been described throughout most of this entry. Although constraints still play a fundamental role (143), now they relate properties of objects (in object databases) to image structure. "Analog," continuous methods, of the sort that have been described throughout, get replaced by more symbolic programming tools (4,144-146,152); for a review, see Ref. 147. It is these symbolic tools that provide the general "inference engine" for interpreting the specific, high level constraints. And these high level constraints are more symbolic than those in early vision; see, e.g., the complex frames and other data structures described in the references above. The mixture of the two can often provide nice solutions to constrained engineering problems (e.g., see Refs. 148 and 149; see also Image understanding).

## Conclusions

Light reflected from physical objects gives rise to images. Vision is the inverse of this process: the recovery of descriptions of objects in the world from images of them. It is clearly an underconstrained problem: somehow a description of 3-D scenes must be recovered from 2-D images. Yet it is possible, as the human visual system demonstrates. But where does the trick lie? How is the structure of the world reflected in the structure of one's visual system? Which aspects of the structure of the world are important, and how are they—should they be—organized? Is it in the gross organization, or in the details of neural interconnections? How can the processing be described so that it could be understood and tested? How does

one's internal percepts relate back to the world? And how can principles be uncovered that allow observations about biological perception to be related to machine perception? These are the kinds of questions that computational-vision research would like to be able to answer.

As shown, there are many ways to approach these questions. Psychology, physiology, anatomy, evolutionary biology, mathematics, computer science, engineering, physics, philosophy, and psychoanalysis all have something to contribute. The diversity of these fields gives some indication of the diversity of constraints that are active in vision, and the goal of this entry has been to illustrate how they can work together. A metaphorical example may help, in retrospect, to put the pieces of this entry together.

Suppose that you were an extraterrestrial who happened to land on this planet in the midst of a museum collection of clocks. That some relationship existed between the objects might be inferred from their proximity, but how might you go about discovering it. Physical and anatomical observations would reveal differences in their microstructure, with some objects composed of sand and others of wood or metal. However, the introduction of the abstract concept of energy would greatly unify the investigation and might even point out some macroscopic principles of organization; namely, that clocks contain (or depend on) a source of energy and have internal structure that acts reliably on the external world. For example, the internal gears might move hands or the transistors control LED displays. To take the next step toward understanding how these devices relate to one another requires another abstract concept: the mathematical notion of periodicity and modular arithmetic. Now consider concepts of time and eventually connect them with the many tasks for which time-keeping is important, ranging from agriculture to social customs.

Of course, the above vignette is grossly oversimplified. The difficulties inherent in making the many leaps are immense. But the point is clear: theoretical ideas from many different levels lead to constraints that percolate via reductionism and constructivism to other levels.

Investigations of the problems of vision rarely yield complete theories. Rather, their contribution results in the formulation of constraints for shaping any theory. Such constraints stand no matter whether the parent theoretical framework changes. The evolution of one's understanding of these constraints has been the principal theme running through this entry; this is probably what is considered to be progress in understanding vision.

## BIBLIOGRAPHY

1. D. Marr, *Vision*, W. H. Freeman, San Francisco, CA, 1982.
2. D. Ballard and C. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
3. R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
4. M. Levine, *Vision in Man and Machine*, Prentice-Hall, Englewood Cliffs, NJ, 1985.
5. T. Pavlidis, *Structural Pattern Recognition*, Springer, New York, 1977.
6. T. Pavlidis, *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville, MD, 1982.

7. R. Nevatia, *Machine Perception*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
8. A. Rosenfeld and A. Kak, *Digital Picture Processing*, Academic Press, New York, 1982.
9. W. Pratt, *Digital Image Processing*, Wiley-Interscience, New York, 1978.
10. R. Gonzalez and P. Wintz, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1977.
11. T. Cornsweet, *Visual Perception*, Academic Press, New York, 1970.
12. R. Gregory, *The Intelligent Eye*, McGraw-Hill, New York, 1970.
13. R. Haber and M. Herschenson, *The Psychology of Visual Perception*, Holt, Rhinehart, and Winston, New York, 1973.
14. L. Kaufman, *Sight and Mind*, Oxford University Press, New York, 1974.
15. I. Rock, *The Logic of Perception*, MIT Press, Cambridge, MA, 1984.
16. W. Uttal, *A Taxonomy of Visual Processes*, Erlbaum, Hillsdale, NJ, 1981.
17. J. Beck, B. Hope, and A. Rosenfeld (eds.), *Human and Machine Vision*, Academic Press, Orlando, FL, 1983.
18. O. Braddick and A. Sleight (eds.), *Physical and Biological Processing of Images*, Springer, New York, 1983.
19. H. von Helmholtz, in J. P. C. Southall (ed.), *Treatise on Physiological Optics*, Dover (reprint), Mineola, NY, 1962.
20. G. Fry, *Blur of the Retinal Image*, Ohio State University Press, Columbus, 1955.
21. S. Coren, L. Ward, C. Porac, and R. Fraser, "The effect of optical blur of visual-geometric illusions," *Bull. Psychon. Soc.* 11(6), 390-392 (1978).
22. L. Roberts, "Machine perception of 3-dimensional solids," in J. Tippett (ed.), *Optical and Electro-Optical Information Processing*, MIT Press, Cambridge, MA, 1965.
23. Reference 22, p. 267.
24. F. Ratliff, *Mach Bands: Quantitative Studies on Neural Networks in the Retina*, Holden Day, San Francisco, CA, 1965.
25. F. S. Werblin, "Functional organization of a vertebrate retina: Sharpening up in space and intensity," *Ann. NY. Acad. Sci.* 193 (1972).
26. R. Kirsch, "Computer determination of the constituent structure of biological images," *Comput. Biomed. Res.* 4, 315-328 (1971).
27. A. Hildebrandt, *Introduction to Numerical Analysis*, Wiley, New York, 1956.
28. A. Herskovitz and T. Binford, On Boundary Detection. AI Memo 183, MIT, Cambridge, MA, 1970.
29. B. Horn, The Binford-Horn Line Finder, AI Memo 285, MIT, Cambridge, MA, 1973.
30. J. M. S. Prewitt, Object Enhancement and Abstraction, in A. Rosenfeld and J. Prewitt (eds.), *Picture Processing and Psychopictorics*, Academic Press, New York, 1970.
31. R. Haralick, and L. Watson, "A facet model for image data," *Comput. Vis. Graph. Im. Proc.* 15, 113-129 (1984); R. Haralick, "Digital step edges from zero crossing of second directional derivative," *IEEE Trans. Pattern Analysis and Machine Intelligence* PAMI-6, 58-68 (1984).
32. M. Heuckel, "An operator which locates edges in digital pictures," *JACM* 18, 113-125 (1971).
33. D. Hubel and T. Wiesel, "Functional architecture of macaque monkey visual cortex," *Proc. Roy. Soc. London B* 198, 1-59, (1977)—a review.
34. R. Rodieck, "Quantitative analysis of cat retinal ganglion cell response to visual stimuli," *Vis. Res.* 5, 583-601 (1965).
35. C. Enroth-Cugell and J. Robson, "The contrast sensitivity of retinal ganglion cells of the cat," *J. Physiol. (Lond.)* 187, 517-552 (1966).
36. H. Wilson and J. Bergen, "A four mechanism model for threshold spatial vision," *Vis. Res.* 19, 19-32 (1979).
37. S. Laughlin, M. Srinivasan, and A. Dubs, "Predictive coding: A fresh view of inhibition in the retina," *Proc. Roy. Soc. London B*, 427-459 (1982).
38. E. H. Land, "The retinex theory of color vision," *Sci. Am.* 237(6), 108-128 (1977).
39. A. Witkin, Scale Space Filtering, in A. Pentland (ed.), *From Pixels to Predicates*, Ablex, Norwood, NJ, 1986, pp. 5-19.
40. D. Marr and E. Hildreth, "Theory of edge detection," *Proc. Roy. Soc. London B* 207, 187-217 (1980).
41. Y. Leclerc and S. W. Zucker, "The local structure of intensity changes in images," *IEEE Trans. PAMI* 9, (1987).
42. R. Watt and M. Morgan, "Mechanisms responsible for the assessment of visual location: Theory and evidence," *Vis. Res.* 23, 97-109 (1983).
43. S. W. Zucker and R. Hummel, Receptive Fields and the Representation of Visual Information, *Human Neurobiology* 5, 121-128 (1986).
44. J. Lettvin, H. Maturana, W. McCulloch, and W. Pitts, "What the frog's eye tells the frog's brain," *Proc. IRE* 47, 1940-1951 (1959).
45. H. Barlow, R. Narasimhan, and A. Rosenfeld, "Visual pattern recognition in machines and animals," *Science* 177, 567-575 (1972).
46. D. Ballard, "Parameter networks," *Artif. Intell.* 22, 235-267 (1984).
47. L. Davis, "Hierarchical generalized Hough transform and line-segment based generalized Hough transforms," *Pattern Recog.* 15, 277-285 (1982).
48. M. B. Clowes, "On seeing things," *Artif. Intell.* 2, 79-116 (1971).
49. M. Minsky and S. Papert, *Perceptions*, MIT Press, Cambridge, MA, 1969.
50. G. Kanisza, *Organization in Vision*, Praeger, New York, 1979.
51. Y. Shirai, Analyzing Intensity Arrays Using Knowledge about Scenes, in P. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975, pp. 93-114.
52. A. K. Mackworth, "Interpreting pictures of polyhedral scenes," *Artif. Intell.* 4, 121-137 (1973).
53. B. Horn, "Understanding image intensities," *Artif. Intell.* 8, 201-231 (1977).
54. S. Shafer, *Shadows and Silhouettes in Computer Vision*, Kluwer Academic, Boston, MA, 1985.
55. R. Woodham, "Analyzing images of curved surfaces," *Artif. Intell.* 17, 17-45 (1981).
56. G. Falk, "Interpretation of important line data as a three-dimensional scene," *Artif. Intell.* 3, 77-100 (1972).
57. K. Turner, *Computer Perception of Carved Objects Using a Television Camera*, Ph.D. Thesis, University of Edinburgh, 1974.
58. L. Davis, "Shape matching using relaxation techniques," *IEEE Trans. PAMI* PAMI-1, 60-72 (1979).
59. H. Freeman, "Computer processing of line drawing images," *Comput. Surv.* 5, 57-97 (1974).
60. J. Tsotsos, "Knowledge of the visual process: Content, form and use," *Pattern Recog.* 17, 13-28 (1984).
61. S. Zucker, A. Rosenfeld, and L. Davis, "General Purpose Models: Expectations about the Unexpected," *Proceedings of the Fourth International Joint Conference Artificial Intelligence*, Tbilisi, Georgia, 1975, pp. 716-720.
62. A. Rosenfeld, "A nonlinear edge detection technique," *Proc. IEEE* 58, 814-816 (1970).

63. D. Marr, "Early processing of visual information," *Proc. Roy. Soc. (London)* **B275**, 483-534 (1976).
64. S. Tanimoto and L. Uhr, *Structured Computer Vision*, Academic Press, New York, 1983.
65. W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.* **5**, 115-133 (1943).
66. J. McCarthy and C. Shannon, *Automata Studies*, Princeton University Press, Princeton, NJ, 1956.
67. S. Winograd and J. Cowan, *Reliable Computation in the Presence of Noise*, MIT Press, Cambridge, MA, 1963.
68. M. Arbib, *The Metaphorical Brain*, Wiley, New York, 1972.
69. D. Hebb, *The Organization of Behavior*, Wiley, New York, 1949.
70. B. Julesz, *Foundations of Cyclopean Perception*, University of Chicago Press, Chicago, IL, 1971.
71. E. Ising, "Contribution to the theory of ferromagnetism," *Z. Physik* **31**, 253-258 (1925).
72. D. Luenberger, *Optimization by Vector Space Methods*, Wiley, New York, 1969.
73. G. Sperling, "Binocular vision: A physical and a neural theory," *Am. J. Psych.* **83**, 461-534 (1970).
74. M. Fischler and R. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. Comput.* **22**, 67-92 (1973).
75. A. Guzman, Decomposition of a Visual Scene into Three-Dimensional Bodies, in A. Grasselli (ed.), *Automatic Interpretation and Classification of Images*, Academic Press, New York, 1969.
76. D. Huffman, Impossible Objects as Nonsense Sentences, in Meltzer and Michie (eds.), *Machine Intelligence*, Vol. 6, Edinburgh University Press, Edinburgh, 1971.
77. D. Waltz, Understanding Line Drawings of Scenes with Shadows, in P. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.
78. A. Rosenfeld, R. Hummel, and S. W. Zucker, "Scene labelling by relaxation operations," *IEEE Trans. Sys. Man Cybernet.* **SMC-6**, 420-433 (1976).
79. R. A. Hummel and S. W. Zucker, "On the foundations of relaxation labeling processes," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-5**, 267-287 (1983).
80. D. Ballard, G. Hinton, and T. Sejnowski, "Parallel visual computation," *Nature* **306**, 21-26 (1983).
81. L. Davis and A. Rosenfeld, "Cooperating processes for low-level vision: A survey," *Artif. Intell.* **17**, 245-264 (1981).
82. W. Kohler, *The Task of Gestalt Psychology*, Princeton University Press, Princeton, NJ, 1969.
83. B. Horn, Obtaining Shape from Shading Information, in P. Winston (ed.), *The Psychology of Computer Vision*, McGraw Hill, New York, 1975, pp. 115-156.
84. A. Pentland, "Local shading analysis," *IEEE Trans. PAMI* **PAMI-6**, 170-187 (1984).
- 84a. A. Pentland, "Fractal based descriptions of natural scenes," *IEEE Trans. PAMI* **PAMI-6**, 661-675 (1984).
85. J. Kender, Shape from Texture, Technical Report, Computer Science Department Carnegie-Mellon University, Pittsburgh, PA, 1980.
86. A. Witkin, "Recovering surface shape and orientation from texture," *Artif. Intell.* **17**, 17-47 (1981).
87. J. Beck, *Surface Color Perception*, Cornell University Press, Ithaca, NY, 1972.
88. J. J. Gibson, *The Perception Of The Visible World*, Houghton Mifflin, Boston, MA, 1950.
89. S. Ullman, "Against direct perception," *Behav. Br. Sci.* **3**, 373-415 (1980).
90. Reference 1, p. 37.
91. H. Barrow and J. M. Tenenbaum, Recovering Intrinsic Scene Characteristics from Images, in A. Hanson and E. Riseman (eds.), *Computer Vision Systems*, Academic Press, New York, 3-26, 1978.
92. W. E. L. Grimson, *From Images to Surfaces*, MIT Press, Cambridge, MA, 1983.
93. D. Terzopoulos, Multilevel Computational Processes for Visible Surface Reconstruction, Ph.D. Thesis, MIT, Cambridge, MA, 1984.
94. J. J. Koenderinck and A. van Doorn, "Photometric invariants related to solid shape," *Opt. Acta* **27**, 981-996 (1980).
95. J. J. Koenderinck and A. van Doorn, "The shape of smooth objects and the way contours end," *Perception* **11**, 129-137 (1982).
96. G. Johanssen, *Configurations in Event Perception*, Almqvist and Wiksells, Uppsala, Sweden, 1950.
97. H. Wallach and D. O'Connell, "The kinetic depth effect," *J. Exp. Psych.* **45**, 205-217 (1953).
98. S. Ullman, *The Interpretation of Visual Motion*, MIT Press, Cambridge, MA, 1979.
99. J. Fodor, *The Modularity of Mind*, MIT Press, Cambridge, MA, 1984.
100. D. Dennett, *Brainstorms*, Bradford Books/MIT Press, Cambridge, MA, 1978.
101. D. van Essen and J. Maunsell, "Two-dimensional maps of the cerebral cortex," *J. Comp. Neurol.* **191**, 255-281 (1980).
102. M. Regan and M. Cynader, "Motion-in-depth neurons; effects and speed and disparity," *Invest. Ophth. Vis. Sci.* **20**, 148 (1981).
103. G. Orban, *Neuronal Operations in the Visual Cortex*, Springer, New York, 1984.
104. S. Zeki and S. Shipp, "Segregation of pathways leading from area V2 to areas V4 and V5 of macaque monkey cortex," *Nature* **315**, 322-324 (1985).
105. H. Barrow and J. M. Tenenbaum, "Computational vision," *Proc. IEEE* **69**, 572-595 (1981).
106. P. Cavanagh, "Reconstructing the third dimension: Interactions between color, texture, motion, binocular disparity, and shape," *Comput. Vis. Graph. Im. Proc.*, **37** (1987).
107. D. Hoffman and B. Flinchbaugh, "The interpretation of biological motion," *Biol. Cybernet.* **42**, 195-204 (1982).
108. J. Cutting, "Coding theory adapted to gait perception," *J. Exp. Psych. Hum. Perc. Perf.* **7**, 71-87 (1981).
109. W. Ittleson, *Visual Space Perception*, Springer, New York, 1960.
110. J. Feldman and D. Ballard, "Connectionist models and their properties," *Cog. Sci.* **6**, 205-254 (1982).
111. S. Ullman, "Visual Routines," *Cognition* **18**, 97-159 (1984).
112. M. Brady and A. Yuille, An Extremum Principle for Shape from Contour, *Proc. of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, 1983.
113. B. Horn and B. Schunk, "Determining optical flow," *Artif. Intell.* **17**, 185-204 (1981).
114. K. Ikeuchi and B. Horn, "Numerical shape from shading and occluding boundaries," *Artif. Intell.* **17**, 141-184 (1981).
115. S. W. Zucker, "Early orientation selection: Tangent fields and the dimensionality of their support," *Comput. Vis. Graph. Im. Proc.* **32**, 74-103 (1985).
116. J. Canny, A computational approach to edge detection, *IEEE Trans. PAMI* **PAMI-8**, 679-698 (1986).
117. L. Quam, Road Tracking and Anomaly Detection, *Proceedings of the DARPA Image Understanding Workshop*, pp. 51-55, 1978.
118. A. Witkin, Intensity Based Edge Classification, *Proceedings of the Second AAAI Conference*, Pittsburgh, PA, pp. 36-41, 1982.

119. K. Stevens, "The visual interpretation of surface contours," *Artif. Intell.* **17**, 47–74 (1981).
120. M. Brady and Asada, "Smoothed local symmetries and their implementation," *Int. J. Robot. Res.* **3**, 36–61 (1984).
121. S. W. Zucker, "On the structure of texture," *Perception* **5**, 419–436 (1976).
122. J. Beck, Texture Segregation, in J. Beck (ed.), *Organization and Representation in Perception*, Erlbaum, Hillsdale, NJ, 1982.
123. R. Haralick, Statistical and Structural Approaches to Texture, *Proceedings of the Fourth International Joint Conference on Pattern Recognition*, Kyoto, Japan, 1978, pp. 45–69.
124. T. Kanade, "Recovery of the three-dimensional shape of an object from a single view," *Artif. Intell.* **17**, 409–460 (1981).
125. G. Dodd and L. Rossol, *Computer Vision and Sensor-Based Robots*, Plenum, New York, 1979.
126. M. Kass and A. Witkin, Analyzing Oriented Patterns, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, 1985, pp. 944–952.
127. D. Hoffman and W. Richards, Parts of Recognition, in A. Pentland (ed.), *From Pixels to Predicates*, Ablex, Norwood, NJ, 1986, pp. 268–294.
128. K. Koffka, *Gestalt Psychology*, Harcourt, Brace and World, New York, 1935.
129. M. Wertheimer, "Laws of organization in perceptual forms," *Psych. Forsch.* **4**, 301–350 (1923); translated in W. Ellis, *A Source Book of Gestalt Psychology*, Routledge and Kegan Paul, London, pp. 71–88, 1938.
130. A. Witkin and J. M. Tenenbaum, On the Role of Structure in Vision, in J. Beck, B. Hope, and A. Rosenfeld (eds.), *Human and Machine Vision*, Academic Press, New York, 1983.
131. D. Lowe and T. Binford, Segregation and Aggregation: An Approach to Figure/Ground Phenomena, *Proceedings of the DARPA Image Understanding Workshop, 1982*, pp. 168–178.
132. D. Lowe, Perceptual Organization and Visual Recognition, Ph.D. Thesis, Stanford University, 1984.
133. J. Mayhew and J. Frisby, "Psychophysical and computational studies towards a theory of human stereopsis," *Artif. Intell.* **17**, 349–385 (1981).
134. D. Kinderlehrer and G. Stampacchia, *An Introduction to Variational Inequalities and their Applications*, Academic Press, New York, 1980.
135. S. W. Zucker and R. A. Hummel, "Toward a low-level description of dot clusters: Labelling edge, interior, and noise points," *Comput. Graph. Im. Proc.* **9**, 213–233 (1979).
136. D. Terzopoulos, Integrating Visual Information from Multiple Sources, in A. Pentland (ed.), *From Pixels to Predicates*, Ablex, Norwood, NJ, 1986, pp. 111–142.
137. T. Binford, Visual Perception by Computer, *Proceedings of the IEEE Conference on Systems and Control*, Miami, 1971.
138. A. Pentland, "Perceptual organization and the representation of natural form," *Artif. Intell.* **28**, 293–331 (1986).
139. A. Barr, "Superquadrics and Angle-Preserving Transformations," *IEEE Computer Graphics and Applications*, 11–23 (1981).
140. J. Foley and A. van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, MA, 1982.
141. A. Triesman, "Preattentive processing in vision," *Comput. Vis. Graph. Im. Proc.* **1**–22 (1985).
142. O. Faugeras, Steps toward a Flexible 3-D Vision System for Robotics, in H. Hanufusa and H. Inoue (eds.), *Robotics Research: The Second International Symposium*, MIT Press, Cambridge, MA, 1985.
143. R. Brooks, "Symbolic reasoning among 3-D models and 2-D images," *Artif. Intell.* **17**, 285–348 (1981).
144. T. Garvey, Perceptual Strategies for Purposive Vision, Technical Note 117, SRI International, Menlo Park, CA, 1976.
145. A. Hanson and E. Riseman, *Computer Vision Systems*, Academic Press, New York, 1978.
146. J. K. Tsotsos, J. Mylopoulos, D. Covvey, and S. W. Zucker, "A framework for visual motion understanding," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-2**, 563–573 (1980).
147. T. Binford, "Survey of model based image analysis systems," *Int. J. Robot. Res.* **1**, 18–64 (1982).
148. F. Ferrie, M. D. Levine, and S. W. Zucker, "Cell tracking: A modeling and minimization approach," *IEEE Trans. Patt. Anal. Mach. Intell.* **4**, 277–291 (1982).
149. M. Levine and S. Shaheen, "A modular computer vision system," *IEEE Trans. PAMI* **PAMI-3**, 540–556 (1981).
150. B. K. P. Horn, Obtaining shape from shading information, in P. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.
151. A. Pentland, Local shading analysis, *IEEE Trans. Patt. Analysis and Machine Inst.* **PAMI-6**, 170–187 (1984).
152. M. Nagao and T. Matsuyama, *A Structural Analysis of Complex Aerial Photographs*, Plenum, New York, 1980.

S. W. ZUCKER  
McGill University

Thanks to the Natural Sciences and Engineering Research Council (Canada), grant A4470, and the Canadian Institute for Advanced Research for their support.

## VISUAL DEPTH MAP

Early vision (qv) is often characterized as a process that reconstructs the three-dimensional properties of scenes from their 2-D images. The dimension lacked by images is the distance along lines of sight from the viewer to points on physical surfaces in the environment. This distance is known as depth.

Humans effortlessly gain a strong sense of depth as they navigate the natural world with both eyes open, and a weaker though definite depth percept results from viewing a monocular image of a 3-D scene. Furthermore, synthetic visual stimuli such as random-dot stereograms and kinetic displays are known to elicit compelling percepts of coherent surfaces in depth.

The recovery of depth from images may be formulated as an inverse problem, converse to the direct problem in computer graphics of rendering images of 3-D geometric models. Evidently, the human visual system is adept at solving this inverse problem at an early processing stage, and it is reasonable to hypothesize the existence of an internal representation of three-dimensionality, putatively in terms of perceived depth  $z$  over the visual field. This representation, naturally expressed as a single valued depth function  $z = u(x, y)$ , where  $x$  and  $y$  are retinotopic or image coordinates, is commonly known as the visual depth map.

The visual depth map has attracted substantial attention in the study of human and machine vision. The first section briefly examines techniques for acquiring initial 3-D information to construct depth maps. These include range sensors engineered to actively acquire depth data from the environment and computational processes of early vision that perform a 3-D analysis of images. In computational vision the visual depth

map has evolved into elaborate representations of the visible surfaces in 3-D scenes. As outlined in the second section, they explicate not only depth but also the orientations, boundaries, and other physical characteristics of visible surfaces. The three subsequent sections examine a difficult and fundamental problem, the reconstruction of visual depth maps from 3-D information provided by multiple visual processes or sensors. In turn, they discuss computational requirements, the inverse mathematical problem inherent to them, and reconstruction algorithms. The final section reviews recent progress on the analysis of visible surfaces described by depth maps to extract surface features that are suitable for object recognition.

### Depth Acquisition

Two approaches may be taken to acquiring depth information from the 3-D environment: active methods, which project energy into the scene, and passive methods, which do not. In modern photography an example of the former approach is ultrasound ranging, whereas using the shallow depth of field of large-aperture lenses to determine distance is an example of the latter.

For the dimensions typically encountered in industrial robotics (qv) applications, active depth sensors based on optical methods seem superior to alternative schemes involving acoustics or radar. Optical depth sensors employ a transmitter, typically a laser, that illuminates the scene and a receiver that captures and processes the light. Pulsed-mode sensors determine depth by measuring the time elapsed from the transmission of a light pulse until its reflection is received. Amplitude-modulated sensors determine depth by measuring the phase difference between the received wave and a reference signal. Triangulation depth sensors employ a structured light source. Usually a spot, line, or grid is projected on a 3-D scene, imaging devices detect the reflected light, and special techniques are used to accurately localize it in the image. Depth can then be estimated accurately by triangulation from the known geometry of the light source and imaging devices. The above three active depth sensors are most common, although other systems have been developed (1) (see Sensors).

Biological vision systems are passive. An array of specialized computational processes exploits generic assumptions about the physical world and image formation to infer the 3-D structure of a scene by interpreting specific 2-D image cues, such as stereoscopic disparity, motion parallax, texture, contours, and shading. Passive shape-estimation processes fall into two broad categories.

The first category comprises what are commonly referred to as correspondence processes. These processes operate over multiple image frames separated by relatively small space-time intervals. Examples are stereopsis (see Stereo vision) and kinetic depth perception (see Motion analysis). Stereopsis is driven by computations on, typically, two image frames simultaneously acquired from different vantage points in space. The basic structure from motion computation involves a temporal sequence of monocular frames usually acquired from a fixed vantage point. If interframe correspondences can be established between image features that derive from the same point on a visible surface (not a trivial problem), then by triangulation, the depth to such points can be estimated from the disparity (or 2-D spatiotemporal displacement) between corresponding features given some knowledge of the imaging geometry.

The second category of shape-estimation processes involves computations on a single static frame. Perspective projection of 3-D scenes onto images induces a systematic distortion of imaged surface properties such as shading, texture, and contours. A major part of this distortion can be attributed to the relative orientations of visible surfaces with respect to the viewer. In principle, it is possible to estimate surface orientation by measuring and interpreting such distortions in the image. This is the basis of practical approaches to recovering surface shape from shading (see Shape analysis), texture (see Texture analysis), and contours (see Scene analysis).

### Depth Representation

The psychologist Gibson (2) made the seminal conjecture that natural perception amounts to the perception of visual surfaces in the environment. This underscored the importance of the retinal projections of visible surfaces and edges of objects as the primary source of 3-D information. Horn was one of the first computer-vision researchers to consider the subproblem of computing continuous depth maps through an analysis of shaded images of continuous surfaces (3).

In addition to depth maps, Horn (3,4) considered the computation of needle maps, which represent local surface orientation. Depth, orientation, and the explicit representation of surface boundaries was suggested by Marr and Nishihara (5-7) in their 2½-D sketch. Barrow and Tenenbaum (8,9) proposed the computation of intrinsic images, a set of maps in spatial registration with the intensity image. Each map represents an intrinsic physical characteristic of the scene, such as depth, orientation, motion, surface reflectance, and incident illumination, together with information about the locations of discontinuities in that characteristic. Intrinsic images are best computed simultaneously, so that well-defined boundary conditions from one map can assist the computation of other maps (10). The multiresolution visible-surface representations proposed by Terzopoulos (11,12) integrate multiple sources of surface depth, orientation, and discontinuity information. This information is fused consistently across multiple scales of spatial resolution.

The perception of visible surfaces is usually immediate and involuntary and seems to precede object recognition. This strongly suggests the existence of a depth-map-reconstruction process that autonomously computes and dynamically maintains depth-map representations. It is necessary for this process to perform at least the following tasks (see below): It must integrate visual information from multiple sources, spatially (and temporally) interpolate visual information from where it is available to where it is lacking, locate discontinuities where interpolation must be limited appropriately, and fuse visual information across multiple scales of resolution.

A computational model of the depth-map-reconstruction process must specify how to perform these tasks in unison and efficiently. Moreover, it ought to be amenable to implementation on highly parallel networks of simple processing elements, computational structures that are suggestive of biological mechanisms. The significant technical challenges posed by these requirements have been satisfied by a computational model that is examined in the next three sections (see also Refs. 11 and 12).

Before this is done, it should be pointed out that several researchers have pursued an alternative approach to visual

recognition. This approach does not attempt to quantitatively reconstruct depth maps of 3-D scenes as a precursor to recognition. Instead, the aim is to infer from images the qualitative 3-D structure of scenes (13–15). The current research base is inadequate for judging whether either approach is sufficient. It suffices to say that, together, these complementary approaches provide a more complete understanding of early visual processing.

### Depth-Map Reconstruction: Computational Requirements

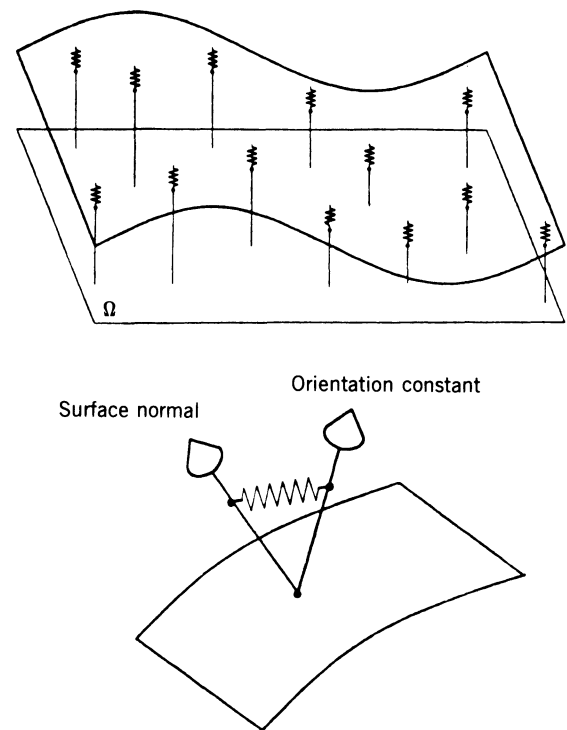
**Multiprocess Integration.** As mentioned above, several visual processes generate initial 3-D information about a scene. To a large extent, each is concerned with computing, from a specific image cue—such as stereoscopic disparity, motion parallax, shading, texture, and contours—information about the geometric and material properties of visible surfaces. These specialized processes appear to be reasonably modular, and a great deal of effort over the last decade was devoted to delimiting modules and studying each independently of the others (16). Although this strategy proved to be fruitful, it evaded certain complex issues raised by what appear to be nontrivial interactions among processes.

Vision researchers quickly came to an important realization: The human visual system integrates diverse processes, each insufficient in itself, into an overall interpretation that is reliable and robust over a wide range of situations. Interactions, particularly the integration of information from multiple visual processes, is now recognized as an essential aspect of early visual processing (see Multisensor integration).

Multiprocess integration is central to the reconstruction of depth maps. In fact, the primary motivation underlying the proposal of representations like the  $2\frac{1}{2}$ -D sketch was that such representations would act as buffers, collecting incoming visual information from earlier processing stages and providing it to later stages. In this context the output of each low-level process may be thought of as a quasi-independent source of information partially constraining the depth map. Integration introduces redundancy, which is necessary not only to resolve potential ambiguities about the map but also to overcome the detrimental effects of noise and inaccuracies in the initial constraints.

Cooperative mechanisms have been devised for integrating multiple sources of surface depth and orientation measurements to reconstruct the shapes of visible surfaces from images (11). The mechanisms are rooted in a mathematical formulation of depth-map reconstruction based on elasticity theory (17).

A physical interpretation of the formulation is illustrated in Figure 1. The depth map  $z = u(x, y)$ , where  $x$  and  $y$  are the image coordinates, may be visualized as an elastic surface whose shape varies smoothly between constraints (except at discontinuities). Constraints deflect the surface from its nominally planar state by imparting forces according to the deformation of a set of ideal springs attached to the constraints. The spring coupling affords a tolerance for inexact constraints. The left part of Figure 1 shows the elastic surface whose deflection  $u(x, y)$  at equilibrium is determined by an infrastructure of scattered depth constraints. Each depth estimate is encoded as the vertical height of the constraint. The influence of each constraint is controlled by the associated spring stiffness: The stiffer the spring, the tighter the constraint. The right part of Figure 1 illustrates the action of an orientation constraint,



**Figure 1.** A physical model for integrating surface depth and orientation data for depth-map reconstruction (13).

which coerces the local surface normal, again via an ideal spring.

Analog models of perceptual processes were prevalent in the school of Gestalt psychology, particularly with regard to the principle of “*Praeganz*,” a minimal-complexity principle inherent to analogue media (18). The above model has a similar character. Visual information from multiple-processes is integrated into the analogue medium (the elastic surface), and the embodied visual representation evolves, subject to the smoothness constraint imposed by the medium, toward a stable state of minimal complexity. In this way potentially contradictory information is resolved, and the stable depth map is as good as the prevailing conditions allow. This is formulated in the language-of-optimization theory (below).

**Interpolation.** Initial representations of images make explicit the occurrence and local 2-D structure of image features that are correlated to salient events on physical surfaces, such as markings, boundaries, etc. This is the role, for instance, of the “*primal-sketch*” representation of significant image-intensity changes or edges (16). Salient features cannot be expected to occur everywhere over the visual field. This implies that surface-shape constraints generated by low-level processes will also be scattered over a subset of image points.

It is fascinating, however, that the human visual system systematically interprets visual stimuli such as sparse random-dot stereograms as surfaces in depth (19). Indeed, these stereograms continue to elicit perceptions of dense surfaces even when the density of dots carrying disparity information has been reduced until depth is unspecified over 98% of the visible surface area.

This phenomenon strongly suggests that the depth-map-reconstruction process is smoothly propagating shape information into indeterminate regions (devoid of shape estimates)



from places where such information is available; i.e., it is "filling in the gaps." In mathematical terms an interpolation of the shape constraints is occurring. Not surprisingly, then, surface-approximation theory can provide a formal basis for interpolating depth maps.

Numerous practical techniques for interpolating and approximating surfaces from scattered data are available [see the reviews by Schumaker (20) and Franke (21)]. These techniques may be classified into local methods, in which any part of the approximating surface depends only on a localized subset of data points, and global methods, where the surface depends on all data points. A powerful global method is based on the so-called generalized splines, a multidimensional generalization of classical splines (22,23). The mathematical theory of generalized splines offers a formal basis for the elastic surface model described earlier.

The second-order generalized spline, known as the thin-plate spline, has been applied widely to interpolation problems in engineering. Thin-plate splines generate surfaces that minimize the functional

$$\iint (u_{xx}^2 + 2u_{xy}^2 + u_{yy}^2) dx dy$$

an integral quadratic measure of the second partial derivatives (denoted by subscripts) of the surface  $u(x, y)$ . This functional is related to the bending energy of a thin flexible plate (24), which characterizes a  $C^1$  smooth surface that is both continuous and possesses continuous derivatives. Another popular instance is the first-order generalized spline, or membrane spline, which minimizes the functional

$$\iint (u_x^2 + u_y^2) dx dy$$

related to the energy of deformation of a stretched membrane (24). It yields a  $C^0$  or continuous surface, which need not have continuous derivatives.

Barrow and Tenenbaum (9,25) suggest an approach to reconstructing smooth depth maps from noisy data that, in essence, minimize integral measures not unlike the generalized spline functionals. They report an implementation of a simple Laplacian relaxation scheme for interpolating the first-partial-derivative functions of the depth map. One can show that Laplacian interpolation of depth derivatives is analogous (aside from boundary conditions) to interpolating the depth map with a thin-plate spline. Subsequently, the thin-plate spline functional was applied directly to the continuous interpolation of visual depth maps from scattered depth constraints by Grimson (26,27) (who refers to the functional as the "quadratic variation"), Terzopoulos (17,28), and Langridge (29). Brady and Horn (30), Blake (31), and Kender et al. (32) provide further analyses of this approach.

**Discontinuities.** Generic continuity assumptions have physical validity inasmuch as the coherence of matter tends to give rise to continuously varying depth maps relative to the viewing distance. However, along with percepts of coherent surfaces, random-dot stereograms elicit vivid impressions of surface discontinuities where disparity changes abruptly. In general, visual discontinuities result from significant, spatially localized changes in the physical world, especially abrupt changes in surface structure.

Discontinuities in depth occur at occluding contours, along which a surface in the scene occludes itself or another surface.

Orientation discontinuities occur at creases or cusps of an otherwise continuous surface. Both depth (or zeroth-order) and orientation (or first-order) discontinuities are perceptually relevant and provide vital boundary conditions for depth-map reconstruction. The depth-map-reconstruction process must find and make explicit surface-depth and orientation discontinuities and enable discontinuities to appropriately limit interpolation.

Generalized splines cannot deal with discontinuities since these functionals offer no spatial control over their continuity properties. In particular, globally smooth interpolation with thin-plate splines destroys surface-depth and orientation discontinuities, an important deficiency of the interpolation schemes implemented in Refs. 17 and 25–28.

A general solution to this problem involves the formulation of controlled-continuity constraints (11,33). These constraints involve generalized splines of multiple orders combined with continuity-control functions. By adjusting the control functions, the smoothness properties of these nonquadratic and spatially noninvariant constraints can be controlled with spatial precision to reconstruct localized discontinuities of any order and arbitrary shape occurring anywhere within the visual field.

The following functional, a controlled-continuity constraint of order 2, has been employed for the depth-map-reconstruction problem:

$$\mathcal{S}_{\rho\tau}(v) = \frac{1}{2} \iint \rho(x, y) \{ \tau(x, y) (v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2) + [1 - \tau(x, y)] (v_x^2 + v_y^2) \} dx dy$$

where  $\rho(x, y)$  and  $\tau(x, y)$  are real-valued continuity-control functions whose range is  $[0, 1]$ . This controlled-continuity constraint is a weighted convex combination of a thin-plate spline and a membrane spline functional. It can be thought of as a thin-plate surface under tension, where  $\rho(x, y)$  is a spatially varying surface "cohesion" and  $[1 - \tau(x, y)]$  is the spatially varying surface "tension" (see Ref. 33 for a discussion).

The local continuity properties of the thin-plate surface-under-tension functional can be controlled at any point  $(x, y)$  by specifying the values of  $\rho(x, y)$  and  $\tau(x, y)$  at that point. As  $\tau$  approaches 1, the functional tends to a thin-plate spline (a  $C^1$  surface), whereas toward the other extreme, 0, the functional tends to a membrane spline (a  $C^0$  surface) with intermediate values characterizing a hybrid  $C^1$  surface that blends the properties of both constituent splines. The function  $\rho$  determines the overall strength of the smoothness functional. Thus, at all nondiscontinuity points  $(x, y)$ ,  $\rho(x, y)$  and  $\tau(x, y)$  are made nonzero. The function  $\tau(x, y)$  is set to zero (or nearly zero) at those points where orientation discontinuities are to be reconstructed. At depth-discontinuity points,  $\rho(x, y)$  is set to zero (or nearly zero).

Discontinuity information known in advance can be introduced into the depth map by presetting the continuity-control functions. In general, however, discontinuities are unknown and must be determined as part of the depth-map-reconstruction process. Two approaches for detection of discontinuities in the depth map have been pursued.

The first approach exploits the fact that, like an image, the depth map is a single-valued function of two variables. Hence, techniques similar to those for detecting intensity discontinuities in images can be adapted to detecting discontinuities in depth maps. Terzopoulos (11) detected depth discontinuities at significant zeros and orientation discontinuities at significant

extrema of the bending moment of the thin-plate spline during the reconstruction process. Langridge (29) employed thin-plate spline curvature peaks to detect surface-orientation discontinuities during reconstruction. Grimson and Pavlidis (34) use the local statistics of the residuals of a local planar approximation to the depth data to detect discontinuities prior to reconstruction. Brady et al. (55) used 1-D scale-space methods (qv) along lines of curvature of a reconstructed surface to find surface discontinuities.

The above methods suffer a disadvantage that is also shared by most image-intensity edge-detection techniques. Since discontinuity detection is sensitive to noise, some sort of local prefiltering is needed. This usually takes the form of smoothing with filters such as Gaussians, locally fitting simple functions such as low-order polynomials, exploiting the smoothing properties of generalized splines, etc. Unfortunately, prefiltering leads to errors, since smoothing smears out discontinuities, making their detection and localization difficult. The second approach aims to avoid these difficulties by directly reconstructing a piecewise smooth function subject to generic constraints on discontinuity structure.

For piecewise-constant image restoration and edge detection, Blake (35) proposes weak constraints that can be broken at a penalty. The constraints are more likely to be broken at the discontinuities in a piecewise-constant function, and the assigned penalty determines the sensitivity to discontinuities. Geman and Geman (36) employ a piecewise-constant Markov random field image model. The model is augmented with a "line process" that penalizes local jump discontinuity configurations according to simple "good continuation" criteria along discontinuities. The criteria have a similar function to those employed by Zucker et al. (37). Marroquin (38,39) uses a more sophisticated Markov field, equivalent to a membrane spline, and a similar line process for depth-map reconstruction. Koch et al. (40) apply a particular type of analog computational model based on "Hopfield networks" in the reconstruction of discontinuities. Terzopoulos (12,33) employs controlled-continuity constraints to reconstruct discontinuities as an integral part of depth-map reconstruction. This poses a distributed parameter-identification problem, where the parameters to be estimated from the data are the continuity-control functions (see below).

**Multiscale Fusion.** There are several important reasons for maintaining depth maps at multiple scales. Psychophysical studies have identified multiresolution spatial frequency channels in the human visual system (41). Their existence has given impetus to the design of low-level visual algorithms that can provide surface-shape constraints at multiple resolutions (16). In addition, machine vision research has demonstrated that multiresolution processing effectively bridges fine and coarse structure, and it simultaneously increases computational efficiency (42) (see Scale-space methods).

Computational efficiency is a critical issue, since depth-map reconstruction at the resolution of the image imposes an immense computational burden. Nevertheless, depth maps must be computed quickly if they are to be of any practical value. The necessary performance may be achieved by emphasizing parallelism in visual algorithms (see Connectionism). Thus, Barrow and Tenenbaum (9,25), Grimson (26,27), and others pursued relaxation or numerical optimization algorithms for surface interpolation. For the most part, these iterative algorithms require only local computations that can be

done in parallel. However, relaxation and optimization are compute-bound, and the fundamental limitations of massively parallel mechanisms, particularly with respect to global interprocessor communications, lead to severe inefficiencies. Excruciatingly slow convergence is observed for interpolation problems of reasonable size.

The depth-map-reconstruction process must not only exploit parallelism but must also overcome cooperative communication bottlenecks to compute depth maps quickly, given suitable architectures. The solution to this problem hinges on the idea of multiresolution structuring of visual representations and associated cooperative processes. Multilevel depth-map reconstruction exploiting multigrid relaxation methods (43) was shown not only to overcome the inefficiency problem but also to be an effective approach to the multiscale fusion of visual information (17,28).

### Depth-Map Reconstruction: Mathematical Aspects

Assuming parallel projection onto the image plane, let the true distance from the viewer to visible surfaces be given by the (unknown) piecewise continuous depth function  $Z(x, y)$ , where  $x$  and  $y$  are the image coordinates. Depth sensing and early visual processing provide a set of noise-corrupted surface-shape estimates (i.e., constraints) that can be expressed in the abstract notation

$$c_i = \mathcal{L}_i(Z(x, y)) + \varepsilon_i$$

where  $\mathcal{L}_i$  denotes measurement functionals of  $Z(x, y)$  and  $\varepsilon_i$  denotes associated measurement errors. The depth-map-reconstruction problem is to compute from the available data  $\{\mathcal{L}_i, c_i\}_{i=1}^N$  a faithful approximation to the depth function  $Z(x, y)$  and an explicit description of its discontinuities.

This inverse problem is made difficult by the nature of the constraints. First, constraints are contributed not by one but by multiple visual processes or sensors. Hence, coincident but slightly inconsistent measurements provided by different processes will locally overdetermine surface shape. Second, constraints are not dense but are scattered sparsely over the visual field. Therefore, although they may restrict surface shape locally, they do not determine it uniquely everywhere; there remain very many feasible depth functions that are consistent with the constraints. Third, the measurements are subject to errors and noise. High spatial frequency noise, no matter how small the amplitude, can locally perturb the surface radically.

In view of the above three considerations, it cannot be assumed that the solution will exist, or that it will be unique, or that it will be stable with respect to measurement errors. Mathematical problems for which these properties cannot be guaranteed a priori are said to be ill-posed (44). Depth-map reconstruction can thus be characterized as a mathematically ill-posed problem; it cannot be solved without applying some additional generic information about possible solutions.

As mentioned above, depth maps are generically constrained due to the physical coherence of visual surfaces. Hence,  $Z(x, y)$  tends to be continuous and differentiable over a substantial part of the visual field. Although continuity is maintained, the derivatives of  $Z(x, y)$  will be discontinuous at surface corners and creases. Finally,  $Z(x, y)$  is entirely discontinuous along occluding contours in the image.

Generic continuity constraints can be readily applied if the depth-map-reconstruction problem is formulated as a varia-

tional principle or optimization problem. The variational-principle formulation can be endowed with the desirable property of being well-posed; i.e., one can guarantee the existence, uniqueness, and stability of the solution under nonrestrictive conditions. This is a systematic approach for solving ill-posed problems, characteristic of the regularization methods introduced by Tikhonov and others (44,45).

There are two parts to the variational principle formulation. The first component is referred to in regularization theory as a stabilizing functional. Denoted symbolically by  $\mathcal{S}(v)$ , the stabilizing functional imposes the generic smoothness assumptions on admissible depth functions  $v(x, y)$ . It stabilizes solutions, making them robust against noise. The second component, denoted by  $\mathcal{P}(v)$ , provides a measure of the discrepancy between admissible depth functions and the available data. It is known as a penalty functional in optimization theory. Simply stated, the variational principle requires finding that admissible depth function  $v = u(x, y)$ , which minimizes  $\mathcal{S}(v) + \mathcal{P}(v)$ .

Assuming independently distributed measurement errors  $\varepsilon_i$  with zero means and variances  $\sigma_i^2$ , the optimal penalty functional is a weighted Euclidean norm of the discrepancy between the admissible function and the data  $c_i$ :

$$\mathcal{P}(v) = \frac{1}{2} \sum_i \alpha_i (\mathcal{L}_i[v] - c_i)^2$$

where the  $\alpha_i$  are nonnegative real-valued weights (ideally  $\alpha_i$  is inversely proportional to  $\sigma_i^2$ ; i.e.,  $\alpha_i = 1/\lambda\sigma_i^2$ ). This penalty functional is optimal given the above assumptions, but it is broadly applicable.

Measurement functionals  $\mathcal{L}_i$  for surface reconstruction may be represented by point evaluation of generalized derivatives. Zeroth-order derivatives, or the evaluation functionals  $\mathcal{L}_i[v] = v(x_i, y_i)$  are sufficient to represent local depth constraints

$$c_i = Z(x_i, y_i) + \varepsilon_i = d_{(x_i, y_i)}$$

First-order derivatives and the vector-valued functional  $\mathcal{L}_i[v] = [v_x(x_i, y_i), v_y(x_i, y_i), -1] = \mathbf{v}_n(x_i, y_i)$  may be used to specify the components of the surface normal  $[Z_x(x, y), Z_y(x, y), -1]$  in representing local orientation constraints:

$$\mathbf{c}_i = [Z_x(x_i, y_i), Z_y(x_i, y_i), -1] + \varepsilon_i = \mathbf{o}_{(x_i, y_i)}$$

It is convenient to separate the constraints into the set  $i \in D$  of image points at which depth constraints  $d_{(x_i, y_i)}$  occur and the set  $i \in O$  at which orientation constraints  $\mathbf{o}_{(x_i, y_i)}$  occur. The penalty functional can then be expressed as

$$\mathcal{P}(v) = \frac{1}{2} \sum_{i \in D} \alpha_{di} [v(x_i, y_i) - d_{(x_i, y_i)}]^2 + \frac{1}{2} \sum_{i \in O} \alpha_{oi} |\mathbf{v}_n(x_i, y_i) - \mathbf{o}_{(x_i, y_i)}|^2$$

where the  $\alpha_i$  parameters are now distinguished as  $\alpha_{di}$  and  $\alpha_{oi}$ . A somewhat more general functional is given in Ref. 11.

The thin-plate surface under tension, mentioned earlier, acts as the stabilizer for depth-map reconstruction. To incorporate discontinuity detection into the variational principle, the stabilizer is written as a functional of the depth-map function  $v(x, y)$  and of the continuity-control functions  $\rho(x, y)$  and  $\tau(x, y)$ :

$$\mathcal{S}(v, \rho, \tau) = \frac{1}{2} \iint \rho(x, y) \{ \tau(x, y) (v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2) + [1 - \tau(x, y)] (v_x^2 + v_y^2) \} dx dy$$

Another stabilizing functional  $\mathcal{D}(\rho, \tau)$  is introduced in conjunction with discontinuities. In its simplest form, the functional can increase monotonically with the total number of discontinuities; e.g.,  $\mathcal{D}(\rho, \tau) = \iint [\beta_d [1 - \rho(x, y)] + \beta_o [1 - \tau(x, y)]] dx dy$ , where  $\beta_d$  and  $\beta_o$  are positive energy-scaling parameters for the depth and orientation discontinuity contributions, respectively. In general, the functional can be designed so as to incorporate generic information about expected discontinuity structure. This information can be encoded in the local energies of a line process (36), as mentioned earlier, or by a 1-D controlled-continuity stabilizer defined on contours (see Refs. 12 and 33 for detailed discussions).

Given the above definitions, the variational principle is then posed as follows: Find  $v = u(x, y)$  along with  $\rho = \rho(x, y)^*$  and  $\tau = \tau(x, y)^*$  that satisfy

$$\min_{\rho, \tau} \mathcal{E}(u, \rho, \tau) + \mathcal{D}(\rho, \tau)$$

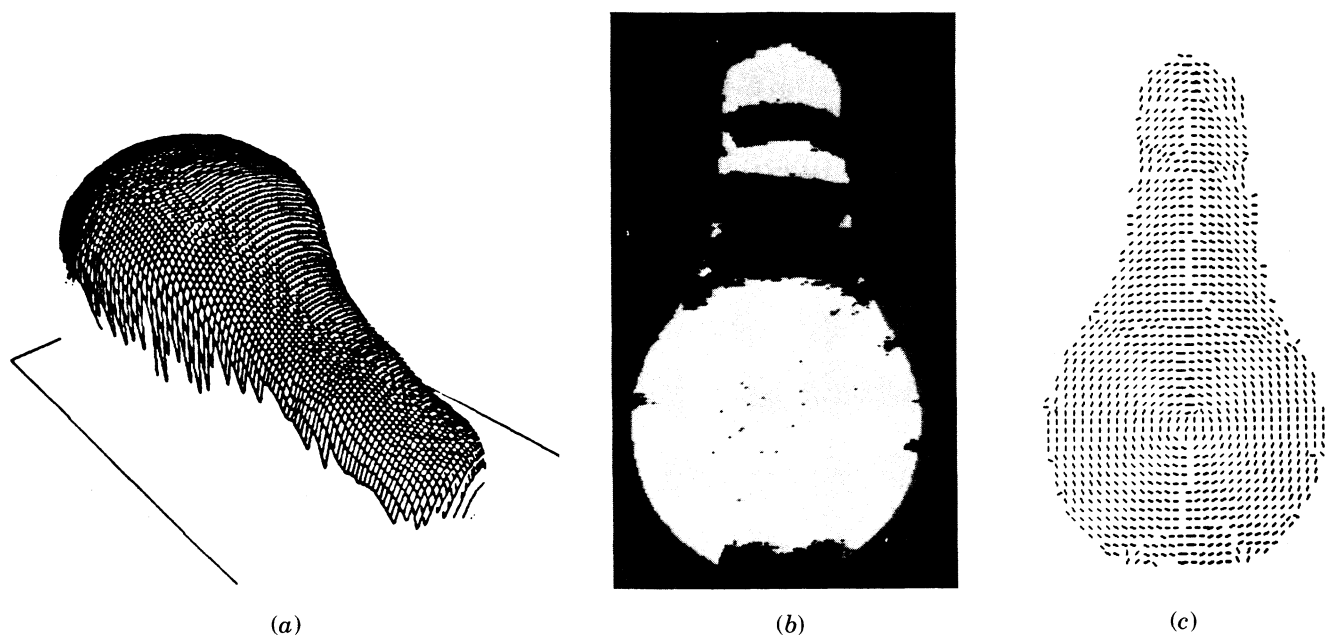
$$\mathcal{E}(u, \rho, \tau) = \min_v \mathcal{S}(v, \rho, \tau) + \mathcal{P}(v)$$

The variational principle involves two minimization problems, one embedded within the other. The internal minimization yields the depth-map function  $u(x, y)$ , given continuity-control parameters (i.e., for prespecified discontinuity structure). The external minimization estimates the continuity-control parameters  $\rho(x, y)^*$  and  $\tau(x, y)^*$ ; i.e., it estimates optimal depth and orientation discontinuity configurations from the data. The internal minimization has a number of strong analytic properties including differentiability and convexity, which imply the existence of a unique solution (11,28). However, there is little hope for the external minimization to have similar properties, since poor analytic structure is characteristic of embedded-optimization problems. The field of nondifferentiable or nonsmooth optimization has recently developed to include such problems, and subgradient optimization is one approach to computing their solutions (46).

### Depth-Map Reconstruction Algorithms

A potent approach to solving the depth-map-reconstruction problem is to express the solution of its variational principle formulation as a superposition-of-basis functions and to numerically compute the unknown coefficients of the superposition. An important consideration is the choice of basis, especially whether it involves local or global support functions. The choice depends critically on the nature of the constraints and discontinuities, and it dictates the character of the reconstruction algorithm.

A global method for computing generalized spline approximations represents the solution as a linear combination of rotationally symmetric, global-support basis functions, one centered at each constraint point (the functions involve fundamental solutions of iterated Laplacian differential equations associated with the variational problem) (22,23). Computing the solution requires first solving a large system of linear equations for the unknown coefficients of the linear combination and then constructing the basis function superposition and restricting it to a compact region of interest in which the solution is continuous. The matrix of the linear system is positive definite, symmetric, and full, and its size is determined by the number of constraints. It can be solved by Cholesky factorization with back substitution, a procedure whose efficiency



**Figure 2.** *a* Reconstructed depth map of a lightbulb; *b* elliptic (white) and hyperbolic (black) points of the reconstructed lightbulb; *c* Principal directions (maximum curvature) for the lightbulb (13).

compares favorably with other direct methods (23). This approach to solving the depth-map-reconstruction problem was pursued by Kender et al. (32) and implemented by Boulton (47).

The alternative is to compute a local support approximation. Terzopoulos (12,17) makes discrete the variational principle for depth-map reconstruction via a local-approximation technique known as the finite-element method (48). The finite-element approximation is a linear combination of local-support basis functions, typically piecewise polynomials of low order. The number of basis functions depends on the number of finite elements that are employed to tessellate the continuous domain. The choice of tessellation is very flexible, but a natural tessellation for depth-map reconstruction follows the image-sampling pattern. With such a tessellation, the size of the resulting positive definite, symmetric, linear system is determined by the desired size of the discrete depth map. Although this is typically greater than the number of constraints, the matrix is sparse due to the local support of the basis functions. This suggests the use of either direct or iterative sparse-matrix techniques to compute the solution.

A critical consideration in favor of the local approach to solving the depth-map-reconstruction problem is the ability to adopt the multigrid-relaxation methods mentioned earlier (17,28). Visual algorithms based on these methods are not only highly efficient but are also natural choices for fusing multi-resolution visual information (49). Finite elements are well suited to multigrid methods and they have several other useful properties. For instance, they are easily defined in the viewer-centered coordinate system that is natural for the incoming visual data, and their free parameters can be made to depend directly on both depth and orientation constraints. More important, the local basis functions associated with finite elements easily enable the introduction and reconstruction of irregularly placed discontinuities; it is not at all obvious how to do this simply with the global method. For more detailed comparisons of the two methods see Refs. 33 and 47.

### Depth-Map Analysis

The depth map is an intermediate, dynamic, and viewpoint-dependent representation of the 3-D surfaces in scenes. It drives ensuing processes that generate stable higher level representations of shape that are better tuned to object recognition. The processing begins with depth-map analysis whose goal is to abstract from the numeric and viewer-centered representation a rich set of more symbolic, object-centered features that are invariant over viewpoint changes. The extraction of geometric surface features is facilitated by the dense, quantitative shape information provided by the depth map.

A promising approach to visible-surface analysis is to apply concepts from differential geometry. For instance, a surface's intrinsic geometry (including Gaussian curvature, geodesics, etc.) is determined completely by the first fundamental form, which defines arc length over the surface. Its extrinsic geometry (including normal curvature, principal curvatures, etc.) is determined by the second fundamental form, which describes the deviation of the surface from the local tangent plane. The fundamental theorem of the local theory of surfaces (usually attributed to Bonnet) states that the analytic study of surface properties consists of the study of the two fundamental forms; i.e., the six fundamental tensor coefficients (which are not all independent) as functions of the two independent parameters of the surface (50). The fundamental forms are invariant under changes in the parameterization, and together they determine surface shape up to rigid-body transformations. These invariance properties make them ideal foundations for object-centered symbolic-surface representations.

The depth-map representation makes it possible to estimate the first and second fundamental forms on a point-by-point basis over the entire visible surface. A finite-element-shape primitive reduces the computation of crucial local surface features such as the Gaussian curvature, principal curvatures, and principal directions to the evaluation of simple algebraic expressions of neighboring values in the depth

map. It is then a simple step to determine the elliptic, hyperbolic, parabolic, umbilic, and planar points as well as geodesics, asymptotes, and lines of curvature.

For example, Figure 2a shows the depth map of a common lightbulb computed by a surface-reconstruction algorithm based on multigrid-relaxation methods, from sparse, noisy data provided by a laser depth sensor (11). The map is rendered in perspective as a 3-D surface bounded by the reconstructed depth discontinuities. Figure 2b shows the Gaussian curvature  $K(x, y)$  computed for the reconstructed lightbulb. The elliptic points ( $K > 0$ ) are shown in white, the hyperbolic points ( $K < 0$ ) are shown in black, and the parabolic points ( $K = 0$ ) separate the two regions. Note the alternating sign of curvature at the screw mount. Figure 2c plots the computed field of principal directions (maximum curvature) for the lightbulb. These demonstrations illustrate the feasibility of reliably computing from visual depth maps useful, higher order intrinsic and extrinsic properties of surface shape. The reliability can be attributed to the stabilizing properties of the thin-plate surface under tension, which overcomes the potentially detrimental effects of noise in the data while preserving discontinuities.

For more detailed investigations into the problem of extracting descriptions of surfaces from depth maps, e.g., see Refs. 11, 12, and 51–55.

## BIBLIOGRAPHY

1. R. A. Jarvis, "A perspective on range finding techniques for computer vision," *IEEE Trans. Pat. Anal. Mach. Intell.* **PAMI-5**, 122–139 (1983).
2. J. J. Gibson, *The Perception of the Visual World*, Houghton Mifflin, Boston, MA, 1950.
3. B. K. P. Horn, Obtaining Shape from Shading Information, in P. H. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, pp. 115–155, 1975.
4. B. K. P. Horn, *Robot Vision*, MIT Press, Cambridge, MA, 1986.
5. D. Marr and H. K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," *Proc. Roy. Soc. Lond. B* **200**, 269–294 (1978).
6. D. Marr, "Visual information processing: The structure and creation of visual representations," *Philos. Trans. Roy. Soc. Lond. B* **290**, 199–218 (1980).
7. H. K. Nishihara, "Intensity, visible surface, and volumetric representations," *Artif. Intell.* **17**, 265–284 (1981).
8. H. G. Barrow and J. M. Tenenbaum, Recovering Intrinsic Scene Characteristics from Images, in A. Hanson and E. Riseman (eds.), *Computer Vision Systems*, Academic Press, New York, pp. 3–26, 1978.
9. H. G. Barrow and J. M. Tenenbaum, "Interpreting line drawings as three-dimensional surfaces," *Artif. Intell.* **17**, 75–116 (1981).
10. B. H. Stuth, D. H. Ballard, and C. M. Brown, Boundary Conditions in Multiple Intrinsic Images, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, FRG, pp. 1068–1072, August 1983.
11. D. Terzopoulos, Multiresolution Computation of Visible Surface Representations, Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, January, 1984.
12. D. Terzopoulos, Computing Visible-Surface Representations, Memo No. 800, MIT AI Lab., Cambridge, MA, 1985.
13. T. O. Binford, "Inferring surfaces from images," *Artif. Intell.* **17**, 205–244 (1981).
14. A. P. Witkin and J. M. Tenenbaum, On the Role of Structure in Vision, in J. Beck, B. Hope, and A. Rosenfeld (eds.), *Human and Machine Vision*, Academic Press, New York, pp. 481–543, 1983.
15. D. G. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer, Boston, MA, 1985.
16. D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W. H. Freeman, San Francisco, CA, 1982.
17. D. Terzopoulos, Multilevel Reconstruction of Visual Surfaces: Variational Principles and Finite Element Representations, AI Memo No. 671, MIT AI Lab, Cambridge, MA; Reprinted in A. Rosenfeld (ed.), *Multiresolution Image Processing and Analysis*, Springer-Verlag, New York, pp. 237–310, 1984.
18. K. Koffka, *Principles of Gestalt Psychology*, Harcourt Brace, New York, 1935.
19. B. Julesz, *Foundations of Cyclopean Perception*, University of Chicago Press, Chicago, IL, 1971.
20. L. L. Schumaker, Fitting Surfaces to Scattered Data, in G. G. Lorentz, C. K. Chui, and L. L. Schumaker (eds.), *Approximation II*, Academic Press, New York, pp. 203–267, 1976.
21. R. Franke, "Scattered data interpolation: Tests of some methods," *Math. Comp.* **38**, 181–199 (1982).
22. J. Duchon, Splines Minimizing Rotation-Invariant Semi-Norms in Sobolev Spaces, in A. Dodd and B. Eckmann (eds.), *Constructive Theory of Functions of Several Variables*, Springer-Verlag, Berlin, pp. 85–100, 1977.
23. J. Meinguet, "Multivariate interpolation at arbitrary points made simple," *Appl. Math. Phys. (ZAMP)* **30**, 292–304 (1979).
24. R. Courant and D. Hilbert, *Methods of Mathematical Physics*, Vol. I, Interscience, London, 1953.
25. H. G. Barrow and J. M. Tenenbaum, Reconstructing Smooth Surfaces From Partial, Noisy Information, in L. S. Baumann (ed.), *Proceedings of DARPA Image Understanding Workshop*, University of Southern California, Los Angeles, CA, Science Applications International Corp., pp. 76–86.
26. W. E. L. Grimson, *From Images to Surfaces: A Computational Study of the Human Early Visual System*, MIT Press, Cambridge, MA, 1981.
27. W. E. L. Grimson, "An implementation of a computational theory of visual surface interpolation," *Comput. Vis. Graph. Img. Proc.* **22**, 39–69 (1983).
28. D. Terzopoulos, "Multilevel computational processes for visual surface reconstruction," *Comput. Vis. Graph. Img. Proc.* **24**, 52–96 (1983).
29. D. J. Langridge, "Detection of discontinuities in the first derivatives of surfaces," *Comput. Vis. Graph. Img. Proc.* **27**, 291–308 (1984).
30. J. M. Brady and B. K. P. Horn, "Rotationally symmetric operators for surface interpolation," *Comput. Vis. Graph. Img. Proc.* **22**, 70–94 (1983).
31. A. Blake, Reconstructing a Visible Surface, *Proceedings of the Fourth National Conference on AI*, Austin, TX, pp. 23–26, August 1984.
32. J. D. Kender, D. Lee, and T. E. Boulton, Information Based Complexity Applied to the 2½-D Sketch, *Proceedings of the Third IEEE Workshop on Computer Vision: Representation and Control*, Miami Beach, FL, pp. 157–167, 1985.
33. D. Terzopoulos, "Regularization of inverse visual problems involving discontinuities," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-8**, 413–424 (1986).
34. W. E. L. Grimson and T. Pavlidis, "Discontinuity detection for visual surface reconstruction," *Comput. Vis. Graph. Img. Proc.* **30**, 316–330 (1985).
35. A. Blake, "The least-disturbance principle and weak constraints," *Patt. Recog. Lett.* **1**, 393–399 (1983).

36. S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-6**, 721-741 (1984).
37. S. W. Zucker, R. A. Hummel, and Rosenfeld, "An application of relaxation labeling to line and curve enhancement," *IEEE Trans. Comput.* **C-26**, 394-403 (1977).
38. J. L. Marroquin, Probabilistic Solution of Inverse Problems, Technical Report 860, MIT AI Lab, Cambridge, MA.
39. J. L. Marroquin, S. Mitter, and T. Poggio, Probabilistic Solution of Ill-Posed Problems in Computational Vision, in L. S. Baumann (ed.), *Proceedings of the DARPA Image Understanding Workshop*, Miami-Beach, FL, Science Applications International Corp., pp. 293-309, 1985.
40. C. Koch, J. L. Marroquin, and A. L. Yuille, Analog "Neuronal" Networks in Early Vision, AI Memo No. 751, MIT AI Lab, Cambridge, MA.
41. O. J. Braddick, F. W. Campbell, and J. Atkinson, Channels in Vision: Basic Aspects, in R. Held, H. W. Leibowitz, and H. L. Teuber (eds.), *Handbook of Sensory Physiology: Perception*, Vol. 8, Springer, Berlin, pp. 3-38, 1978.
42. A. Rosenfeld (ed.), *Multiresolution Image Processing and Analysis*, Springer-Verlag, New York, 1984.
43. W. Hackbusch and U. Trottenberg (eds.), *Multigrid Methods*, Lecture Notes in Mathematics, Vol. 960, Springer-Verlag, New York, 1982.
44. A. N. Tikhonov and V. A. Arsenin, *Solutions of Ill-Posed Problems*, Winston and Sons, Washington, DC, 1977.
45. T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature* **317** (6035), 314-319 (1985).
46. F. H. Clarke, *Optimization and Nonsmooth Analysis*, Wiley-Interscience, New York, 1983.
47. T. E. Boulton, Visual Surface Interpolation: A Comparison of Two Methods, in L. S. Baumann (ed.), *Proceedings of the DARPA Image Understanding Workshop*, Miami Beach, FL, Science Applications International Corp., pp. 466-478, 1985.
48. G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ, 1973.
49. D. Terzopoulos, "Image analysis using multigrid relaxation methods," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-8**, 129-139 (1986).
50. M. P. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Englewood Cliffs, NJ, 1976.
51. P. J. Besl and R. C. Jain, "Invariant surface characteristics for three-dimensional object recognition in range images," *Comput. Vis. Graph. Proc.* **33**, 33-80 (1986).
52. T. J. Fan, G. Medioni, and R. Nevatia, Description of Surfaces from Range Data, in L. S. Baumann (ed.), *Proceeding of the DARPA Image Understanding Workshop*, Miami Beach, FL, Science Applications International Corp., pp. 232-244, 1985.
53. G. Medioni and R. Nevatia, Description of 3D Surfaces using Curvature Properties, in L. S. Baumann (ed.), *Proceedings of the DARPA Image Understanding Workshop*, New Orleans, LA, Science Applications International Corp., pp. 291-229, 1984.
54. B. C. Vemuri, A. Mitiche, and J. K. Aggarwal, 3-D Object Representation from Range Data using Intrinsic Surface Properties, in T. Kanade (ed.), *3-D Vision Systems*, Kluwer Academic, Hingham, MA, 1986.
55. J. M. Brady, J. Ponce, A. Yuille, and H. Asada, "Describing surfaces," *Comput. Vis. Graph. Proc.* **32**, 1-28 (1985).

## VITERBI ALGORITHM

The Viterbi algorithm is a pattern-recognition (qv) technique to assign a sequence of possibly distorted (or noisy) patterns to a sequence of symbolic classes. In deciding the symbolic classification of each pattern it allows taking into account the context of other patterns in the sequence and their symbolic assignments. Its application, most often, is in the recognition of distorted patterns in image and speech recognition (see Image understanding; Speech recognition). The method can be thought of as a shortest path algorithm for a graph whose structure has the form of a trellis, i.e., the nodes of the graph are arranged as stages (Fig. 1). The Viterbi algorithm is a dynamic programming solution to the optimization problem of minimizing or maximizing the cost of traversing the trellis from the leftmost node to the rightmost node. The method was originally formulated as a decoding algorithm in signal processing (1), a field in which it has found a large number of applications (see the survey paper in Ref. 2). More recently, it has found several applications in search problems commonly encountered in machine perception (3-6).

The recognition problem is mathematically formulated as follows. The observed sequence of patterns is

$$\mathbf{X} = X_0 X_1 \cdots X_m X_{m+1}.$$

Each pattern  $X_i$  is to be assigned to one symbol in the set  $\mathbf{L} = \{L_1, L_2, \dots, L_n\}$ . Thus, there are  $r^{m+2}$  possible symbol sequences that may be assigned to the pattern sequence. The merit of any single assignment

$$\mathbf{Z} = Z_0 Z_1 \cdots Z_m Z_{m+1},$$

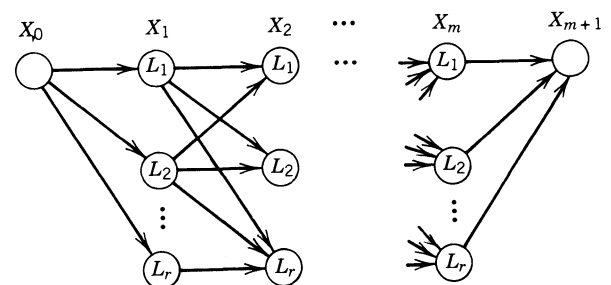
where each  $Z_i \in \mathbf{L}$ , is determined by the criterion

$$C(\mathbf{X}, \mathbf{Z}) = \sum_{i=0}^{m+1} f_1(X_i, Z_i) + \sum_{i=0}^m f_2(Z_i, Z_{i+1}) \quad (1)$$

where  $f_1(X_i, Z_i)$  is a measure of the confidence of assigning pattern  $X_i$  to symbol  $Z_i$  and  $f_2(Z_i, Z_{i+1})$  is a measure of the confidence of  $Z_{i+1}$  occurring after  $Z_i$ . The objective is to determine the sequence  $\mathbf{Z}$  that maximizes the criterion  $C(\mathbf{X}, \mathbf{Z})$ .

### The Algorithm

In terms of the trellis (Fig. 1),  $X_0$  and  $X_{m+1}$  are assumed to be perfectly recognizable. Each of the other  $X_i$  has  $r$  nodes associated with it where each of the nodes corresponds to an element



**Figure 1.** The Viterbi algorithm finds the minimum-cost path through a trellis.



of  $\mathbf{L}$ . There is an edge between each node of stage  $i$  (corresponding to  $X_i$ ) and each node of stage  $i + 1$ . The  $j$ th node ( $1 \leq j \leq r$ ) of stage  $i$  is associated with the negative cost  $-f_1(X_i, L_j)$  and the edge between the  $j$ th node of stage  $i$  and the  $k$ th node of stage  $i + 1$  is associated with the negative cost  $-f_2(L_j, L_k)$ . Thus, the number of distinct paths between  $X_0$  and  $X_{m+1}$  is  $r^m$ , each path representing a distinct symbol sequence. The cost of a path is the sum of all the node and edge costs contained in the path. The problem is one of finding the least cost path.

The Viterbi algorithm involves the evaluation of only  $(m - 1) \times r^2$  paths for  $m > 1$ . This is done by keeping track of two items at each node of the current stage  $i$ : the least cost of reaching that node and the least cost path consisting of a list of  $i - 1$  symbols from  $\mathbf{L}$  (called the survivor) to that node. Given this information for stage  $i$ , the corresponding information for stage  $i + 1$  is computed by considering all the  $r^2$  pairs of nodes between the two stages (only  $r$  pairs between stage 0 and 1 and between stage  $m$  and  $m + 1$ ). The least cost of reaching a node  $j$  ( $1 \leq j \leq r$ ) of stage  $i + 1$  is computed as the minimum, over all nodes  $k$  ( $1 \leq k \leq r$ ) of stage  $i$ , of the following sum: the least cost of reaching node  $k$  plus the cost of reaching node  $j$  from node  $k$ . At node  $j$  is stored the least cost of reaching node  $j$  and the survivor obtained by concatenating a symbol at the end of the appropriate survivor from stage  $i$ . Once all nodes of stage  $i + 1$  have been evaluated, the data for stage  $i$  can be discarded and the same processing is repeated for the next stage. When the algorithm terminates, the survivor at stage  $m + 1$  is the desired sequence, and its cost is also available.

### Probabilistic Criteria Using a Bayesian Formulation

The theoretical justification for a criterion of the form of Eq. 1 can be found in Bayesian decision theory. Given the pattern sequence  $\mathbf{X}$ , the a posteriori probability that the state of nature is the symbol sequence  $\mathbf{Z}$  is given from Bayesian decision theory (qv) as

$$P(\mathbf{Z}|\mathbf{X}) = \frac{P(\mathbf{X}|\mathbf{Z}) \times P(\mathbf{Z})}{P(\mathbf{X})}$$

where  $P(\mathbf{X}|\mathbf{Z})$  is the conditional probability of observing  $\mathbf{X}$  when  $\mathbf{Z}$  is the true sequence,  $P(\mathbf{Z})$  is the a priori probability of  $\mathbf{Z}$ ,  $P(\mathbf{X})$  is the probability of sequence  $\mathbf{X}$ . If  $P(\mathbf{X})$  is assumed to be independent of  $\mathbf{Z}$ , then the sequence  $\mathbf{Z}$  that maximizes  $P(\mathbf{X}|\mathbf{Z})$  can be determined by maximizing the expression

$$G(\mathbf{X}|\mathbf{Z}) = \log P(\mathbf{X}|\mathbf{Z}) + \log P(\mathbf{Z}) \quad (2)$$

Storing the  $P(\mathbf{X}|\mathbf{Z})$  distribution in memory is usually impractical due to the large number of combinatorial possibilities for both  $\mathbf{X}$  and  $\mathbf{Z}$ . Assuming conditional independence among the elements of  $\mathbf{X}$  leads to considerable simplification. According to this assumption, the observed patterns are independent of each other (e.g., in the case of printed text the shapes of letters are not usually influenced by the shapes of adjacent letters). With this assumption,

$$\log P(\mathbf{X}|\mathbf{Z}) = \sum_{i=0}^{m+1} \log P(X_i|Z_i) \quad (3)$$

where  $P(X_i|Z_i)$  is the probability of observing symbol  $X_i$  when the true symbol is  $Z_i$ ; it is sometimes known as the confusion probability.

If one assumes that symbol sequences are generated by an  $n$ th-order Markov source, the a priori probability  $P(\mathbf{Z})$  can be expressed as

$$P(\mathbf{Z}) = P(Z_{m+1}|Z_{m+1-n} \cdots Z_m) \cdots P(Z_1|Z_0) \times P(Z_0)$$

where  $P(Z_k|Z_{k-n} \cdots Z_{k-1})$  is called the  $n$ th-order transitional probability, i.e., the probability of observing  $Z_k$  when the previous  $n$  symbols are  $Z_{k-n} \cdots Z_{k-1}$ . In the case of  $n = 1$ ,

$$P(\mathbf{Z}) = P(Z_{m+1}|Z_m) \cdots P(Z_1|Z_0) \times P(Z_0) \quad (4)$$

Substituting Eqs. 3 and 4 in Eq. 2 yields the criterion

$$G(\mathbf{X}, \mathbf{Z}) = \sum_{i=0}^{m+1} \log P(X_i|Z_i) + \sum_{i=0}^m \log P(Z_{i+1}|Z_i) \quad (5)$$

Both Eqs. 5 and 1 have the same general form, and thus the Viterbi algorithm can be applied to find the symbol sequence that has the maximum a posteriori probability.

### Heuristic Modifications

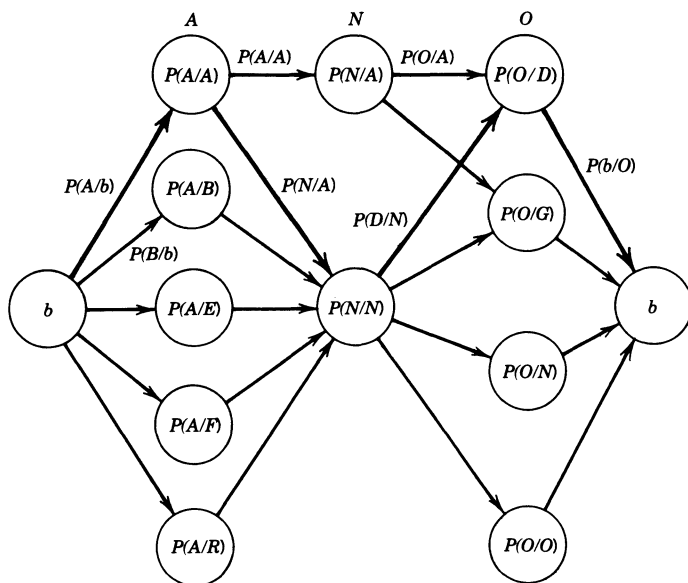
The Viterbi algorithm is a method of finding the exact symbol sequence  $\mathbf{Z}$  that maximizes  $C(\mathbf{X}, \mathbf{Z})$  over all possible  $\mathbf{Z}$ . Its complexity varies linearly with the length of the input sequence  $m$  and quadratically with the size of the alphabet  $r$ . Efficiency of the algorithm can be significantly improved by using heuristic (see Heuristics) modifications that almost always find the most likely sequence.

The simplest modification is to use a fixed number of  $d$  ( $d \ll r$ ) alternatives for each  $X_i$ , where the  $d$  alternatives are the most likely ones for the pattern (3). A second improvement is to use a judiciously chosen variable number of alternatives for each  $X_i$  (5). The use of a variable number of alternatives is illustrated in the following section.

### Application to Spelling Correction and Text Recognition

In the problem of spelling correction, the patterns and symbols are both English letters. In the text-recognition problem the patterns correspond to images (or features) of letters, and symbols correspond to identities of letters in words. The trellis is illustrated in Figure 2 for spelling correction using the 10 letter ( $r = 10$ ) alphabet  $L = \{A, B, D, E, F, G, N, O, R, Y\}$ . The input pattern sequence with  $m = 3$  is the word **bANOb**, where **b** indicates a delimiting blank. The 10 alternatives for each input letter are sorted by letter-confusion probability, i.e., the probability of substituting a letter by another letter, and only those that exceed a threshold  $t$  are considered. The trellis shows the information used in deciding the output string when there are five, two, and four alternatives that exceed a threshold for the letters A, N, and O, respectively. Each of the nodes (except the start and end nodes) and the edges have values associated with them. The node values are confusion probabilities and the edge values are transitional probabilities. Any path from the start node to the end node represents a sequence of letters; note that by considering a variable number of alternatives, one has 18 possible path evaluations instead of 200 path evaluations by the Viterbi algorithm and 1000 path evaluations by the brute-force approach.

It is often necessary to find the most likely sequence in a



**Figure 2.** Heuristic modification of the Viterbi algorithm to use a variable number of alternatives for each input variable. The pattern sequence is the word ANO and the dark path corresponds to the symbol sequence AND.

legal subset of the  $r^m$  possible sequences, i.e., a dictionary. Organization of the dictionary in the form of a tree, called trie, is particularly suitable for use with the Viterbi algorithm (5). In a trie, symbol sequences with identical initial subsequences (prefixes) share the same path in the trie. The Viterbi algorithm can use this data structure to select only those alternatives at stage  $i$  that can correspond to legal paths in the trie.

The performance of utilizing dictionary information during the classification of symbols in this manner has been shown to be superior to the two-step approach of determining the most likely symbol sequence and then correcting the result to fit a dictionary entry (6).

## BIBLIOGRAPHY

1. A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theor.* **IT-13**, 260–269 (1967).
2. G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE* **61**, 268–278 (1973).
3. R. K. Shinghal and G. Toussaint, "The modified Viterbi algorithm for text recognition," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-1**(4), 184–192 (1979).
4. H. Tanaka, Y. Hirakawa, and S. Kanku, "Recognition of distorted patterns using the Viterbi algorithm," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-4**(1), 18–25 (1982).
5. S. N. Srihari, J. J. Hull, and R. Choudhari, "Integrating diverse knowledge sources in text recognition," *ACM Trans. Ofc. Inform. Syst.* **1**(1), 68–87 (1982).
6. J. J. Hull, S. N. Srihari, and R. Choudhari, "An integrated algorithm for text recognition: Comparison with a cascaded algorithm," *IEEE Trans. Patt. Anal. Mach. Intell.* **PAMI-5**(4), 384–395 (1983).

S. SRIHARI  
SUNY at Buffalo

**VORONOI TESSELATION.** See Dot-pattern analysis; Texture analysis.

# W

**WAGERING GAMES.** See Game-playing systems.

## WALTZ FILTERING

"Waltz filtering" is a special form of "constraint propagation," sometimes termed "Boolean constraint propagation" (see Constraint satisfaction). Waltz filtering is a method for simplifying certain combinatorially explosive tree-search problems (see Search). The method uses constraints to reduce the size of the tree to be searched. For certain types of problems Waltz filtering can totally or nearly totally eliminate the need for tree search. The method arose in solving a computer vision (qv) problem, the analysis of line drawings of scenes (1–3). The general type of algorithm was first described by Fikes (4) in 1970.

### Labeling Line Drawings

Waltz's specific analysis problem was to assign a label (chosen from a finite set of possible labels) to each edge of a line draw-

ing, where each label gives semantic information about the nature of the edge and the regions on each of its sides. Figure 1 has been labeled: the labels show that the edge  $v_1-v_2$  is an outside edge (0) of an object, and that it makes a concave edge (–) with its support surface; edge  $v_3-v_4$  is a shadow edge; edge  $v_2-v_{13}$  is a convex (+) edge with both bounding regions visible; edge  $v_1-v_{14}$  is an outside (0) convex (+) edge; etc. The dots near the edges mark the object region for outside edges, and the shaded side of a shadow edge. (This figure uses a simplified version of Waltz's label set, which distinguished 57 different edge labels).

Labeling was originally proposed as a method for line-drawing analysis by Guzman (5), whose work inspired independent articles by Huffman (6) and Clowes (7). All used a very simple world consisting of only trihedral polyhedra, with no shadows and no aligned object faces that could lead to "cracks." (Trihedral polyhedra are objects consisting of plane faces, where exactly three planes meet at any vertex. Thus cubes, prisms, and tetrahedrons are trihedral objects, but a pyramid is not, since four planes meet at its apex.) In the world of trihedral polyhedra, assuming that there are no shadows

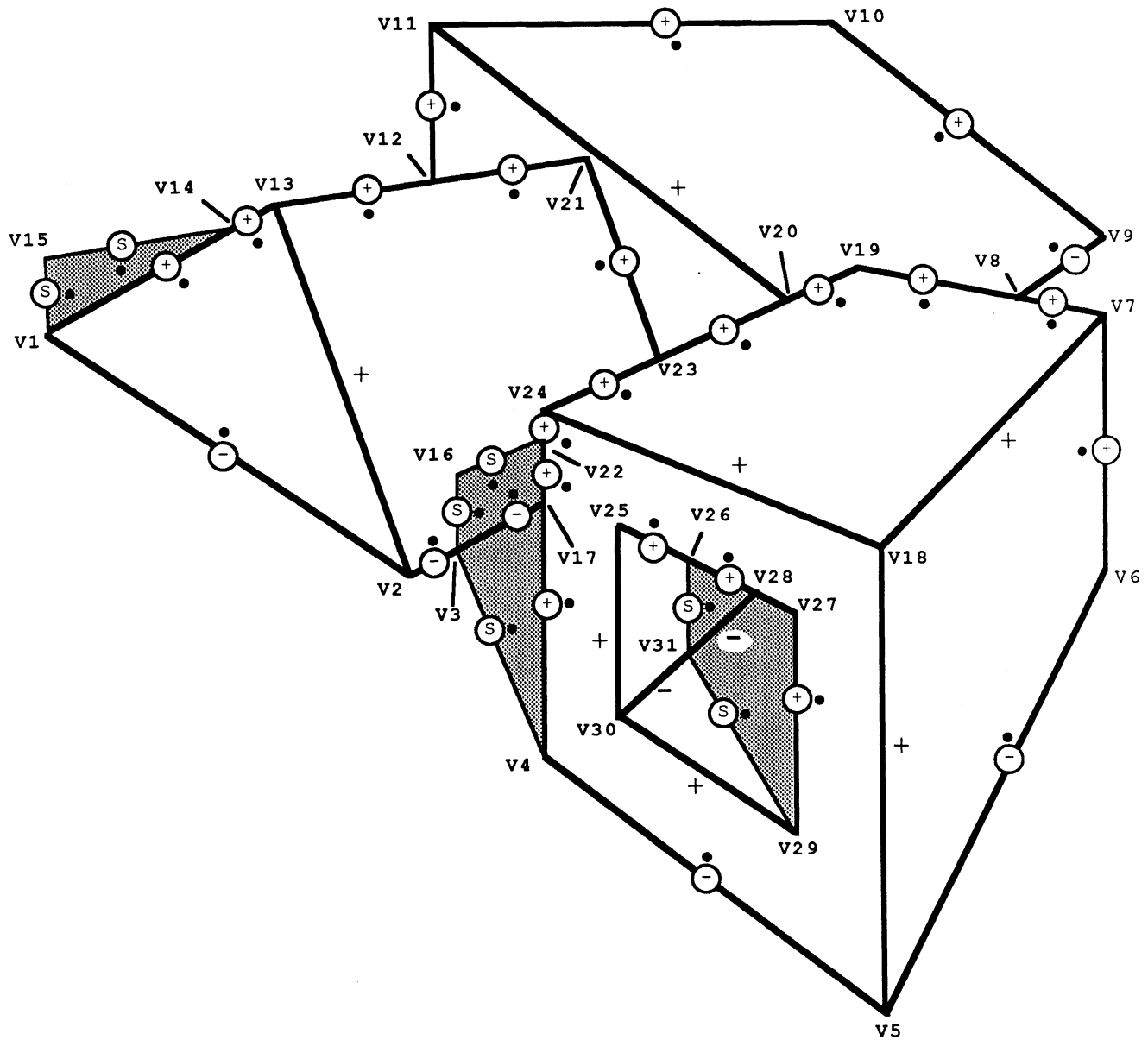


Figure 1. A labeled scene.

and no alignments (visual or actual) of vertices or edges, only a very small number of 2-D vertex types can occur in a line drawing (2-D projection) of a scene. All the allowable vertices in the Huffman-Clowes world are shown in Figure 2. (Huffman and Clowes did not distinguish concave edges that occur at the base of an object resting on a surface from concave edges where both surfaces are part of the same object.) The vertex labels constrain the labels of the edges and constitute all the "knowledge" available in this problem. As illustrated, there are six labels for L-type vertices, four for ARROWS, three for FORKS, and four for T's.

The problem of labeling a line-drawing graph was formulated as a "consistent labeling" problem: Assume that each vertex  $v_i$  has associated with it a set of potential labelings  $L_i$ . The consistent labeling problem is to assign a labeling  $l_{ik} \in L_i$  to each vertex  $v_i$  such that each edge  $e_{ij}$  is assigned the same label by  $l_{ik}$  and  $l_{jm} \in L_j$ , where  $v_i$  and  $v_j$  are the two vertices that define  $e_{ij}$ .

Huffman and Clowes proposed solving the consistent label-

ing problem by a tree-search procedure. In such a procedure one starts by selecting some vertex,  $v_i$ , and assigning it a particular labeling  $l_{i1} \in L_i$ . One then picks another vertex,  $v_j$ , and assigns it a labeling,  $l_{j1} \in L_j$ . Assume that  $v_i$  and  $v_j$  are picked so that  $v_i$  and  $v_j$  share an edge,  $e_{ij}$ . If  $l_{i1}$  and  $l_{j1}$  both assign the same label to  $e_{ij}$ , one can continue assigning labelings to other vertices, depth-first, checking that each new labeling agrees with all previous ones. If  $l_{i1}$  and  $l_{j1}$  assign different labels to  $e_{ij}$ , one must try other assignments for  $v_j$ ; if none are left, one must backtrack, trying different labeling assignments for  $v_i$ . Eventually, this procedure will find all possible consistent labelings. One or more consistent labelings may result (see Fig. 3), but it is possible that no consistent labeling can be found (as in the "impossible object" in Fig. 4).

Tree search is a process with exponential complexity. However, small trees may still be tractable. The cost of this kind of labeling procedure can be estimated for some typical cases: If there are  $p$  possible labels for each edge, the chance that two arbitrarily labeled vertices will share a common label for an

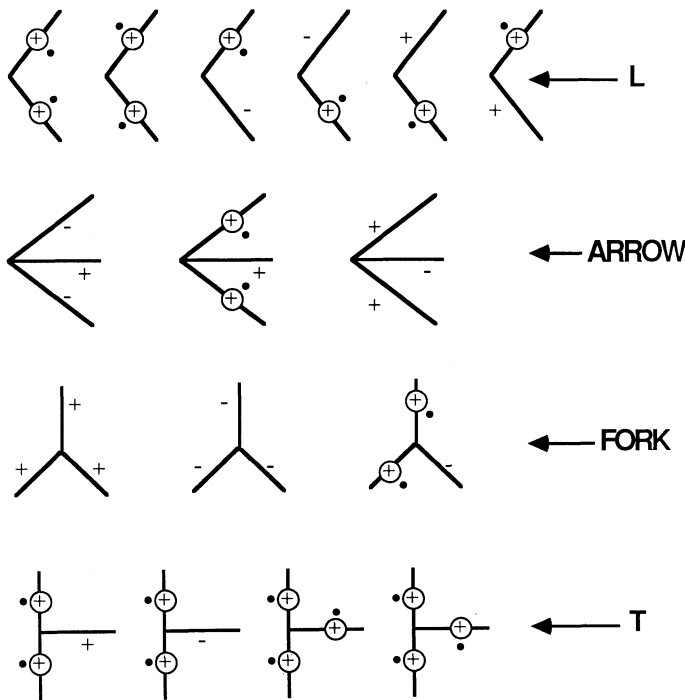


Figure 2. All the labeled vertices for the world of trihedral polyhedra.

edge is  $1/p$ . In scenes of trihedral polyhedra the average number of edges that intersect at each vertex is 3. Assume there are an average of  $m$  labelings possible a priori for each vertex. The total tree size for a scene with  $n$  vertices is  $m^n$ . However, because the tree is pruned when labels fail to match, one will have to visit fewer nodes. To find all labelings, one will have to visit  $m$  nodes for the first vertex,  $m/p$  for the second vertex, and  $m/p^3$  for the last set of vertices to be added. Thus, roughly, the search should involve less than  $m(m/p)^{n-1}$  nodes, assum-

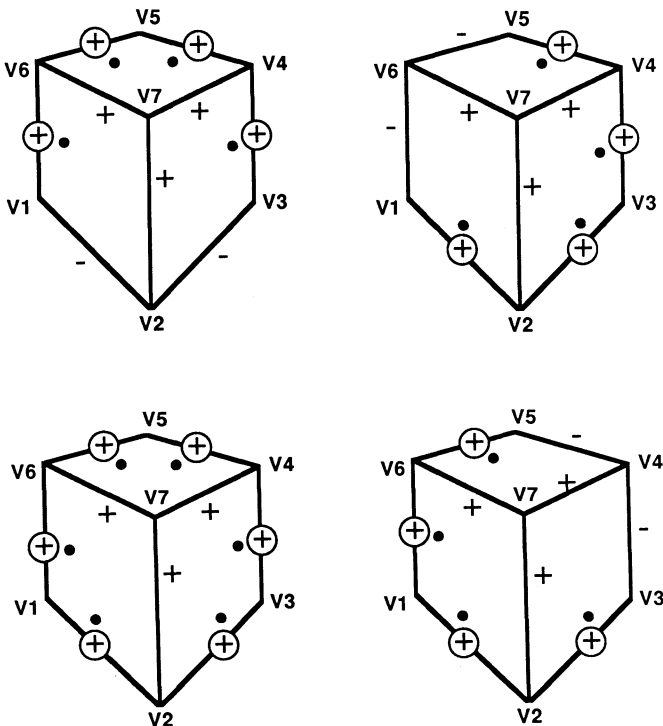


Figure 3. Four ways of labeling a line drawing of a cube.

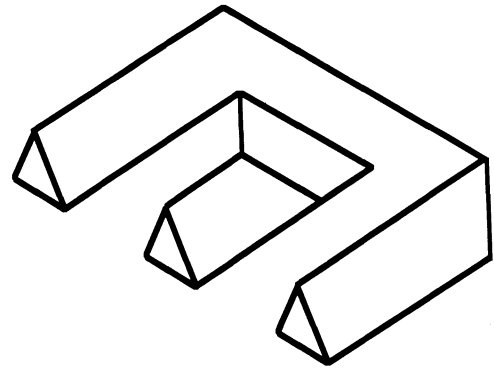


Figure 4. Cannot be labeled and can thus be shown to not correspond to a real trihedral object.

ing that one always adds vertices adjacent to previously labeled vertices and, therefore, that the true average branching factor of the search tree is  $m/p$  or less.

For the Huffman and Clowes world of polyhedral objects,  $m \approx 4.5$ , and  $p = 4$ . This bounds the search tree size at  $4.5 \times (1.125)^{n-1} \approx 160$  nodes for a 31-vertex scene. [Note that the scene in Figure 1 has 31 vertices but cannot be labeled by the Huffman-Clowes set. This is because there are shadows and also "psi" vertices (e.g.,  $v_3$  and  $v_{31}$ ), which are not included in the Huffman-Clowes world. (Psi refers to the shape of the junction that resembles the Greek letter psi.)] This means that even fairly large scenes are tractable.

For more realistic scenes, including shadows and cracks (edges where two aligned objects meet), the numbers are not favorable. The set of labels developed by Waltz for this more complex world had 57 different edge labels and an average of 350 possible labels for any particular vertex type (i.e.,  $p = 57$  and  $m \approx 350$ ). Thus, for the same 31-vertex scene, the search tree in this case could contain as many as  $57 \times (350/57)^{30} \approx 1.14 \times 10^{25}$  nodes.

### Waltz Filtering

To make this problem more tractable, Waltz attempted to cut down  $m$ , the average number of labels that would have to be considered at each vertex. Since edges must be assigned the same label from both ends, one can remove from consideration any vertex labels that fail to have corresponding labels at adjacent vertices. This was accomplished by looking at all the labels  $L_i$  possible for vertex  $v_i$  and all the labels  $L_j$  possible for vertex  $v_j$ ; forming  $S_i$ , the set of possible labels for edge  $e_{ij}$  when viewed from  $v_i$ , and  $S_j$ , the set of all possible labels for the edge  $e_{ij}$  when viewed from  $v_j$ ; finding the intersection of  $S_i \cap S_j$ ; and then only retaining labels in  $L_i$  and  $L_j$  that assign labels to  $e_{ij}$  that are in  $S_i \cap S_j$ .

The procedure just described only compares labels for a pair of vertices. In general, the effects of eliminating labels at one vertex may propagate to a number of other vertices. Suppose that vertex  $v_i$  has been compared with  $v_j$  and all labels  $e_{ij}$  that do not match have been removed. Now consider comparing  $v_i$  with another neighbor,  $v_k$ . If any labels that do not match for  $e_{ik}$  are eliminated for  $v_i$  and  $v_k$ , one ought to reconsider the set of labels assigned to  $v_j$ ; it may be that some more can now be removed from  $v_j$ 's set, since their matches may have been removed from  $v_i$ . This removal of vertex labelings by pairwise comparison and propagation was termed "filtering" by Waltz;

later authors have used the terms "constraint propagation" and "constraint squeezing" for the same process.

In the polyhedral world devised by Waltz, the filtering process led to a remarkable result: For many scenes only one label remained at each vertex after the "filtering" operation, so no tree search at all was necessary.

The cost of the filtering process is impossible to estimate precisely because it depends very heavily on the order in which vertices are considered: If the first vertex considered has a small number of possible labels, it will tightly constrain its neighbors, which will in turn tightly constrain their neighbors, etc. Several vertices types had fewer than 50 labels each, guaranteeing fast performance if one started with them.

In any case, the complexity is no longer exponential: To propagate effects to all other vertices from a particular vertex is at worst  $O(n)$ , so the overall process is at worst  $O(n^2)$ . Actual measurements of elapsed compute time suggested that the actual complexity was close to  $O(n)$ . At the time it was indicated that this was because the propagation of effects was in fact constant, not  $O(n)$ , because effects rarely propagated past  $T$  vertices. Roughly speaking, this is because  $T$ 's are generally the result of one object occluding another; and the labeling of one object has no effect on the object it occludes. Subsequent analysis by Mackworth and Freuder (8) showed that the true complexity is indeed linear.

#### Assessment of Waltz Filtering for Various Problems

Because it is often helpful to use Waltz filtering on a tree-search problem, the process does not always eliminate the need for tree search. For example, it does not eliminate the need for tree search in the Huffman-Clowes polyhedral world. The addition of many more label types made the result possible. To see this, consider the number of labels for particular vertices. In the Huffman-Clowes world there are three possible labels for an ARROW-type vertex. If there were no constraints, each of the three edges that meet at the ARROW could be labeled four ways, so there would be  $4^3 = 64$  possible ways to label the edges. Thus,  $3/64$ , or 4.7%, of the a priori labels for edges are actually possible. This is a measure of the "tightness" of the constraint. In the Waltz world there are about 125 labels for an ARROW, out of  $57^3$ , for a correspondingly far greater "tightness" of  $125/185193 = 0.067\%$ .

Gaschnig (9-11) and Freuder (12) investigated conditions under which Waltz filtering can reduce tree search. Unfortunately, it does not work for all problems. Freuder (13) showed how to extend the process to  $n$ -ary constraints rather than pairwise constraints. Mackworth (14) provided a formal analysis of this process.

The Waltz-filtering process can be easily implemented in parallel on a Connection Machine (qv) (15) by assigning a set of processors to each vertex and edge; each vertex gets as many processors as there are possible labels for the vertex, and each edge gets as many processors as there are edge types. Connections are formed between edge-label processors and their matching vertex-label processors. One can then alternate the following two steps until no more processors are removed by a step:

Each edge label processor that is only connected to one or more vertex-label processors at one of its ends should send a message to those vertex-label processors to remove themselves, and the edge-label processor should then mark itself inactive.

Each vertex-label processor that receives a removal message should remove all its connections and mark itself inactive.

#### BIBLIOGRAPHY

1. D. L. Waltz, Generating Semantic Descriptions from Drawings of Scenes with Shadows, Technical Report AI-TR-271, MIT Artificial Intelligence Laboratory, Cambridge, MA, November 1972.
2. D. L. Waltz, Understanding Line Drawings of Scenes with Shadows, in P. H. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, pp. 19-91, 1975.
3. D. L. Waltz, Automata Theoretical Approach to Visual Information Processing, in R. T. Yeh (ed.), *Applied Computation Theory: Analysis, Design, Modeling*, Prentice-Hall, Englewood Cliffs, NJ, pp. 468-529, 1976.
4. R. E. Fikes, "REF-ARF: A system for solving problems stated as procedures," *Artif. Intell.* 1(1-2), 27-120 (1970).
5. A. Guzman, Computer Recognition of Three-Dimensional Objects in a Visual Scene, Technical Report AI-TR-228, MIT Artificial Intelligence Laboratory, Cambridge, MA, December 1968.
6. D. Huffman, Impossible Objects as Nonsense Sentences, in B. Meltzer and D. Michie (eds.), *Machine Intelligence*, Vol. 6, Edinburgh University Press, Edinburgh, UK, pp. 295-323, 1971.
7. M. Clowes, "On seeing things," *Artif. Intell.* 2(1), 79-116 (1971).
8. A. K. Mackworth and E. C. Freuder, "The complexity of some polynomial network consistency algorithms for constraint satisfaction problems," *Artif. Intell.* 25(1), 65-74 (1984).
9. J. A. Gaschnig, Constraint Satisfaction Method for Inference Making, *Proceedings of the Twelfth Annual Allerton Conference Circuit and System Theory*, University of Illinois, Urbana-Champaign, pp. 866-874, 1974.
10. J. A. Gaschnig, A General Backtrack Algorithm that Eliminates Most Redundant Tests, *Proceedings of the Fifth International Conference on Artificial Intelligence*, Cambridge, MA, August 1977, p. 457.
11. J. A. Gaschnig, Performance Measurement and Analysis of Certain Search Algorithms, Thesis, CMU-CS-79-124, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1979.
12. E. Freuder, "A sufficient condition for backtrack-free search," *JACM* 19, 24-32 (1982).
13. E. Freuder, "Synthesizing constraint expressions," *CACM* 21, 958-966 (1978).
14. A. K. Mackworth, "Consistency in networks of relations," *Artif. Intell.* 8(1), 99-118 (1977).
15. W. D. Hillis, *The Connection Machine*, MIT Press, Cambridge, MA, 1985.

D. L. WALTZ  
Thinking Machines Corp.  
and Brandeis University

# X

---

## XCON

A rule-based system (qv), sometimes called R1, that configures VAX-11/780 computer systems, XCON was written in 1980 by McDermott at Carnegie-Mellon University (see J. McDermott, "R1: A rule-based configurer of computer systems," *Artif. Intell.* **19**, 39-88 (1982)).

M. TAIE  
SUNY at Buffalo

# Z

---

**ZERO CROSSINGS.** See Edge detection; Region-based segmentation; Stereo vision; Vision, early; Visual depth map.



# INDEX

Volume 1, text pages 1–679  
Volume 2, text pages 680–1166

- A\* algorithm, 1
  - best-first search, 997, 999
  - bidirectional search, 1000
  - branch-and-bound search, 1003
  - evaluation functions, 379
  - knowledge engineering, 297
  - Manhattan Distance heuristic, 378
  - means-ends analysis, 579–580
  - mobile robots, 960
  - path planning, 710
  - problem solving, 773–774
  - properties, 1
  - reasoning, AND/OR representation, 825–826
  - route search, 43
  - SHAKY robots, 1039
  - solution path, 1
- AARON programs, 8–9
- ABACUS system, 485
- Abduction, concept learning, 188
- ABEL program:
  - causal reasoning, 828
  - medical advice, 585
- Abelson, R.:
  - frame theory, 302, 304
  - story understanding, 828
- Absity system:
  - knowledge-base semantics, 1028–1029
  - situation semantics, 1026–1027
- Absolute value, automatic programming construction, 33
- Abstract data types, logic programming, 556
- Abstract profiles, inductive inference methods, 414
- Abstraction space, ABSTRIPS program, 582
- ABSTRIPS system:
  - means-ends analysis, 578, 582
  - problem solving, 778
- Ackerman, P. L., 431–439
- Acoustic-phonetic analysis, 1078–1079
- Acoustic ranging sensors, 39
  - obstacle detection, 40
- Acquaintances, object-oriented programming, 452
- ACRONYM system:
  - generalized cylinders, 323
  - image understanding, representational formalisms, 399–401
  - industrial robots, 952
  - knowledge engineering, 297
  - shape analysis, 1046–1047
- ACT (Adaptive control of thought) system, 113
  - language acquisition, 146
  - machine learning, 485
- ACT\* learning theory, 157
- Act3 actor language, 2
- Action:
  - belief systems, 63
  - conceptual dependency rules, 195–196
  - hermeneutics, 368
    - dramaturgical, 367
    - instrumental and strategic, 367
    - normatively regulation, 367
  - intelligent agents, 837
  - meta-reasoning, 598
  - path planning and obstacle avoidance, 708
  - planning, 748–750
    - domain-dependent criteria, 753–754
    - selection, 753–754
  - primitives, 196–198
  - intelligent agents, 837
  - speech acts, 1063
  - units (AU), IPP (integrated partial parser), 699
  - see also specific types of action*
- ACTION system, frame theory, 307
- Activation:
  - goal-directed, 395–396
  - mechanisms, cognitive modeling, 112–114
  - model-directed, 396
- Active lexical interaction:
  - word-expert parsing, 702
  - formalization, 703–704
- Actor formalism, 2
  - control structures, 213
  - distributed problem solving, 246
  - transaction, 2
- Actors:
  - causal reasoning, 828
  - demons, 232
  - logic programming, 556–557
  - object-oriented programming, 452, 455
  - script, 981
- AD-HAC system, expectation-driven parsing, 699
- Ada language:
  - industrial robots, 950
  - inheritance hierarchies, 422
  - limits of, 501
  - recursion, 875
  - SIMULA and, 1048
- Adaptation, intelligence and, 432
- Adaptive Control Implementation, robot arm dynamics, 919–920
- Adaptive robot programs, 39
- Addanki, S., 200–205
- Adjacency matrix, obstacle detection, 40
- Advice Taker, 3, 519
- AEGIS system, 611
- Aerial photo interpretation, stereo vision, 1083
- AGE system:
  - blackboard architecture, 73, 77–78
  - domain knowledge, 252
  - knowledge engineering, 297
  - rule-based systems, 971
- Agenda-based systems, 3–4
  - AI systems, 3–4
  - benefits, 4
  - KRL system, 888
  - natural-language automatic programming, 32
  - rule-based systems, 968
- Agent-object rule, language acquisition, 448–449
- Agglomerative techniques, clustering, 105
- Aggregation:
  - knowledge representation, 886
  - logic programming, 549
  - uncertainty, 854
    - conjunctions and disjunctions, 860
- Agre, P., 211–217
- Agricultural chemicals, 96
- Ahuja, N., 254–256, 1101–1111
- AI-ED communication list, 533
- Aikins, J. S., 3–4
- AILIST mailing list, 533
- AIMDS language, 459
- AIRPLAN knowledge system, 297–298
- Akl, Selim G., 88–93
- AL language for prostheses, 799
- Albus, James, 1054
- Alexis personal vehicle, 800
- Alfred P. Sloan Foundation, 120
- Algebra techniques for chemistry, 96
- Algebraic syntax, 510–511
- ALGOL language:
  - automatic programming synthesis, 24
  - computational linguistics, 133

- control structures:
  - programming language design, 212
  - virtual machines, 213
- objected-oriented programming, 456
- POP-2, 758
- Algorithms:
  - A\*, 1–2
    - best-first search, 997
    - iterative deepening, 998
    - knowledge engineering, 297
    - problem solving, 773–774
    - reasoning, 825–826
  - alpha-beta pruning, 5
    - computer chess, 161–162
  - AO\*, 8
    - best-first search, 999
    - branch-and-bound search, 1003
    - heuristic search, 765–766
  - Ag, 471
  - B\*, 1003
  - back-propagation learning demons, 232
  - best-first search, 998–999
  - bidirectional heuristic front-to-front (BHFFA), 1000
  - bracket abstraction, 442
  - brain's computing power, 490
  - branch-and-bound search, 1003
  - branching factor, 81–82
  - British Museum, 457
  - Canny's edge detection, 263–264
  - case frame, 670–671
  - classifier, inheritance hierarchy, 427
  - clustering, 104
  - collision-detection, 711
  - computational:
    - complexity and, 492
    - transformational grammar, 354
  - computed torque, 917
  - constraint satisfaction:
    - consistency, 207–209
    - cooperation, 205–206
  - contextual postprocessing, 83
  - control structures, 211
  - depth-first search, 1004–1005
  - depth map reconstruction, 1157–1158
  - dictionary viterbi (DVA), 86
  - DLPA, obstacle avoidance, 42
  - Earley's, 325
  - early vision, 1145
    - controlled hallucination, 1137
  - edge detecting, feature acquisition, 1086
  - edge-following, path planning, 710
  - fast Fourier transform, 576
  - feature extraction, 301
  - feedback, robot arms, 904
  - flowchart synthesis, 22–25
  - free-space, path planning, 710
  - Freuder's constraint satisfaction, 209
  - generalized cylinders, 323
  - graph-theoretic, 256
  - heuristics, 378
  - iterative-deepening A\*, 998
  - Knuth-Bendix completion, 1120–1121
  - language learnability models, 446
  - learning demons, 232
  - left-corner parsing, 690
  - linear:
    - motion analysis, 622
    - optical flow, 626
  - logic programming, expert systems, 554
  - material requirements planning, 37
  - Metropolis constraint satisfaction, 210
  - motion analysis accuracy, 629–630
  - multipass coroutines, 220
  - multiple inheritance, 423–424, 426
  - parsing, 327
    - ATN grammar, 330
    - music applications, 638
  - pattern matching, 716
  - PD control schemes, 904
  - polynomial time, 414–416
  - problem reduction, 765
  - program synthesis, 28–29
  - programming, 269
    - reading, 83
  - recursive-descent parsing, 689–690
  - region splitting, 878
  - relaxation:
    - constraint satisfaction, 209–210
    - feature extraction, 301
    - revised therapy, 589–590
    - route finding, 866
    - route planning, mobile robots, 960
    - search, inheritance hierarchy, 424
    - shape-from-texture, 1111
    - Shapiro synthesis, 27–28
    - simulation, causal reasoning, 830
    - smoothness assumption, 686
    - split-and-merge, 879
    - SSS\*, 82, 1003
      - pattern recognition, 726
    - surface orientation, 1042–1043
    - symbolic relaxation, 208
    - test-selection, medical advice systems, 588
    - unification, inductive inference, 413
    - Viterbi, 1160–1162
    - Waltz filtering, 208
    - Wang's, 562
  - ALICE graph-reduction machine, 555
  - Alker, H., 371
  - Allen, B., 309
  - Allen, J. F., 1062–1070
    - speech act theory, 68–69
  - Allen, Jonathon, 1065–1076
    - speech recognition, 1065–1070
    - speech synthesis, 1070–1076
  - Allophone:
    - beam search, 57
    - speech understanding, 1078
  - ALPAC system, machine translation, 141
  - Alpha-beta pruning, 4–6
    - alpha cutoff, 5–6
    - beta cutoff, 5–6
    - branch-and-bound search, 1003
    - branching factor, 82
    - checkers programs, 89–90
    - computer chess, 161–162
      - searching/pruning, 165–166
      - transposition, 164–165
    - search, game playing, 165–166, 316
  - Alto machine, computer architecture, 216
  - ALU in connection machines, 200
  - Alvarado, S., 980–992
  - ALVEN system, image understanding, 396–397, 399–400, 404, 408
  - AM system, 7
    - agenda-based system, 3–4, 848
    - concept learning, 187
    - creativity, 224
    - EURISKO program, 287
    - problem solving, 778
    - qualitative laws, 484
  - Amarel, S., 767–778
  - AMBER language:
    - language acquisition, 145–146, 447
    - sentence-meaning pairs, 481
  - Ambiguity:
    - lexical, expectation-driven parsing, 699
    - natural-language understanding, 661
    - qualitative physics, 810
    - semantic memory, 593
    - word-sense, word-expert parsing, 702
  - Ambiguous networks in inheritance hierarchy, 424–425
  - American Association for Artificial Intelligence (AAAI), 16, 1060
  - Ames, C., 638–641
  - AML language, industrial robots, 948
  - AMORD system, dependency-directed backtracking, 48
  - Amsler, R. A., 530–534
  - Analogical reasoning, problem solving, 767
  - Analogue representation, 881
  - Analogy:
    - game-playing, 312
    - learning by, 15, 476–479
      - derivational method, 477–478
      - geometry, 476–477
      - means-end analysis, 476
      - transformational analogy, 476–477
  - Analysis:
    - depth-map, 1158–1159
    - dot-pattern analysis, 253–256
    - Greek concept, 775
    - means-ends, 578–584
    - morphological, 620
    - motion, 620–630
      - accuracy of algorithms, 629–630
      - flow fields, 626–627
      - future research, 630
      - high-level motion understanding, 630
      - image intensity matching, 627–628
      - modeling and prediction, 630
      - multiple objects, 630
      - nonrigid objects, 630
      - optical flow motion, 624–625
      - orthographic projections, 624
      - planar curve correspondence, 624
      - 3-D feature correspondences, 628–629
      - three-view case, 623–624
      - two-view motion with feature correspondences, 621–624
        - point correspondence, 621–622
  - ANALYST program, 606–608
  - Analytic philosophy, hermeneutics, 363
  - Anaphora reference:
    - discourse, 239
    - ellipsis, 272–273
    - natural-language understanding, 658, 672
    - procedural semantics, 1030
    - semantic grammar, 352
    - semantic networks, 1020
  - Anaphora resolution:
    - artificial intelligence and, 14–15
    - discourse understanding, 144
  - Anderson, J. R., 443

- AND/OR graph, 7–8  
 applications, 7  
 best-first search, 999  
 branch-and-bound search, 1003  
 game trees, 319–321  
 image understanding, 400  
 pattern recognition, 726  
 problem reduction, 762–763, 995  
 search, 764–766  
 problem solving:  
 reduction schema, 774  
 SAINT system, 973  
 search, 7–8  
 systemic grammar, 653  
 trees, reasoning, 825
- Angluin, D., 409–416
- Annealing, atomic system, 204
- Anomalies, inductive inference, 415
- Anthropomorphic robots, 944–954  
 current research, 949–950
- Antibehaviorism, 737
- Antimechanism, Godel argument, 739–740
- AO\* algorithm, 8  
 best-first search, 999  
 branch-and-bound search, 1003  
 heuristic search, 765–766
- Apel, K., hermeneutics, 366, 369–370
- APES (Augmented-PROLOG expert system), 553
- Aphasia, 506
- Apiary network, actor formalisms, 2
- APL, semantic networks implementation, 1022
- Aplysia californica*, 489
- Apollo II moon rocks, ATNs, 324
- Apple Macintosh, 512, 803
- APPLIER system, conceptual dependency, 987
- Apply goal, means-ends analysis, 579
- Appropriate query assumption in question answering, 819
- APSG (Augmented phrase structure grammar), 143
- AP3 relational database system, 78
- APT language, industrial robots, 951
- Aq algorithm, 471
- AQ11 system, 191, 485
- Arc consistency algorithm, constraint satisfaction, 208–209
- Arch concept  
 analytic approach to concept learning, 471–473  
 machine learning, 465–466
- Arch-learning program (Winston), 741
- Architecture  
 hierarchical, speech understanding, 1080  
 programming tools, 794  
 von-Neumann computer systems, 174
- Arc-pair grammar, 354
- Arcs, ATN parsing, 693
- Area matching, stereo vision, 1087
- AREA system, LISP machine, 529
- ARGOS system, beam modeling, 397–398
- ARGOT system, office automation, 140
- Aristotle:  
 cybernetics, 226  
 epistemology, 281
- Arithmetic:  
 clustering rules, 110  
 Godel's theorem, 739–740  
 intelligence and, 432
- Arora, K. S., 81–82, 312, 598, 632, 677, 687, 746, 1049
- ARPA project:  
 phonological theory, 1079  
 speech understanding in system architecture, 1080
- ARPA Speech Understanding Research Program, 384
- ARPANET program, 526
- AI literature, 533
- Art  
 AI in, 8–9, 223  
 belief revision, 58
- Artificial belief system (ABS), 70
- Artificial computer languages, computational linguistics, 133
- Artificial Intelligence*, 120
- Artificial intelligence (AI), 9–16, 740  
 applications, 13–14  
 in chemistry, 93–97  
 Church's thesis, 99–100  
 cognitive modeling and, 114  
 cognitive psychology, 116  
 cognitive science, 120–123  
 completeness, 132  
 computer-aided design (CAD), 152–153  
 computer-integrated manufacturing, 172–173  
 computer systems for, 174  
 corouting, 222–223  
 educational applications, 267–271  
 office automation, 681  
 empiricism, 742  
 epistemology, 284–285  
 fully structured material, 491  
 functionalism, 737–738  
 future development, 1050–1051  
 game playing, 312–318  
 hermeneutics, 362–363, 370–373  
 history, 9–10  
 human-computer interaction, 383–385  
 information retrieval, 420–421  
 knowledge representation and inference, 12–13  
 languages and machines, 15–16  
 lambda calculus, 442  
 law applications, 456–461  
 learning, 14–15  
 limits of, 488–501  
 brain as biochemical computer, 488–489  
 brain's "programs", 490–491  
 computational complexity, 491–492  
 moral limits, 502  
 philosophical issues, 488  
 present state of knowledge, 492–501  
 AI methodology, 500–501  
 motor control, modeling of spatial environments and motion planning, 496–497  
 reasoning, planning, knowledge representation and expert systems, 497–501  
 sensory functions, 493–496  
 programming languages, 500–502  
 automatic programming, 501–502  
 quantitative estimates of brain's computing power, 489–490
- literature, 530–534  
 anti-AI literature, 532  
 beginnings, 530  
 cognitive science, 531–532  
 expert systems, 533  
 fifth generation computers, 533  
 future trends, 533–534  
 historical literature, 532  
 LISP literature, 532  
 logic programming and PROLOG, 532–533  
 1960s, 531  
 1970s, 531–532  
 1980s, 532–533  
 overviews, 532  
 pre-AI literature, 530
- logic and, 537  
 machine translation, 564–566, 568–570  
 in medicine (AIM), 584  
 military applications, 604–613  
 political issues, 1056–1057  
 monotonic reasoning, 13  
 music and, 638–641  
 myths about, 493  
 nonmonotonic reasoning, 13  
 phenomenology and, 731–735  
 philosophical questions, 736–743  
 physical symbol system hypothesis, 9  
 political implications of, 502  
 problem solving and search, 10–12  
 programming environment, 789–790  
 propositional logic, 562  
 rule-based systems, 967  
 social issues, 1049–1059  
 strong, 537  
 temporal reasoning, 871–872  
 weak, 537
- Artificial Intelligence Corporation, 139
- Artificial Intelligence in Medicine*, 82
- Artificial Intelligence in Model Building (AIMB) program, 96
- ASK system, computational linguistics, 135, 138
- Aspects model, transformational grammar, 692
- Aspects of the Theory of Syntax*, 134, 356–360, 443–444
- Aspiration search:  
 computer chess, 162  
 software advances, 166–167
- Assembly language, LISP interpreter, 518
- Association, cognitive modeling, 111–112
- Association for Computing Machinery (ACM), 1060
- Association-oriented approach, medical diagnosis, 454
- Associative memory, 16–18  
 associative disks, 17  
 information retrieval, 421  
 memory management and hardware control, 17  
 outlook, 17  
 semantics, 593  
 set search, 17  
 STARAN, 17
- Assumption-based belief systems, 60–61
- Assumptions:  
 propositional logic deduction systems, 562  
 scale-space imaging, 976–977

- ATMS, belief revision, 58
- ATN grammar, 323-332  
 INTELLECT system, 431  
 language acquisition, 443-445  
 LUNAR system, 694-695  
*see also Augmented transition network and Grammar, augmented transition network*
- Atomics, propositional logic, 559
- Atoms, LISP system, 509
- ATTENDING system, 585-586  
 augmented transition networks, 590  
 research issues, 588-589
- Attention:  
 change and focus, image understanding, 393  
 cognitive psychology, 116
- Attribute values, cluster analysis, 107
- Augmented decision network, medical advice systems, 585
- Augmented transition network (ATN), 323-332  
 cognitive modeling, 113  
 complexity and nonmodularity, 667  
 computational linguistics, 136  
 deep structure, 231  
 fragility, 667  
 grammar, 142-143, 323-332  
   cascaded, 331-332  
   expectation-driven parsing, 696  
   factoring, 325  
   formal properties, 329  
   generalized transition networks, 331  
   history, 325-326  
   linguistic experimentation, 328-329  
   misconceptions about, 330-331  
   parsing, 327, 330  
   perspective, 329-330  
   self-looping arcs, 330-331  
   specification, 327-328  
 grammatical deviation, 672  
 inefficiency:  
   through backtracking, 667  
   through meaningless parses, 667  
 language acquisition, 443  
   linguistic theory, 444-445  
   sentence-meaning pairs, 480-481
- LAS program, 480
- logical formalisms and context dependence, 239
- LUNAR system, 694-695  
 machine translation, 565  
 medical advice systems, 588-589  
 natural language generation, 646-647  
 parsing, 693-694  
   AD-HAC system, 699  
   speed and efficiency, 694-695
- PLANES, 748
- semantic networks, 1021
- speech synthesis, 1074
- speech understanding, 1079
- text generation, 144-145
- transformational grammar and natural language understanding, 666  
*see also ATN grammar and Grammar, augmented transition network*
- AURA program, binary resolution, 900
- Autoepistemic reasoning, 851
- AUTOLING program, language acquisition, 145-146
- Automata theory, 530
- speech understanding, 1079
- Automated decision making, military applications, 605-606
- Automated Manufacturing Research Facility (AMRF), 949
- Automated natural language understanding, military applications, 612-613
- Automated tutoring system, GUIDON, 362
- Automatic guided vehicles (AGVs), 39
- Automatic Language Processing Advisory Committee (ALPAC), 133
- Automatic programming, 18-34  
 assistants, 785  
 computational linguistics, 140-141  
 examples, 18  
 history, 22  
 input-output specifications, 19-22  
   deductive mechanism, 20-21  
   stating the problem, 19-20  
 limits of, 501-502  
 mechanized assistant construction, 32-34  
   program maintenance and improvement, 34  
 program synthesis, 21-22  
   examples, 22-29  
   flowchart construction revision, 24-25  
   flowcharts from example traces, 22-23  
   historical remarks, 29  
   LISP code, 25-26  
     recurrence relations, 26  
   natural language dialogue, 29-32  
   PROLOG programs, 27-28  
   theoretical issues, 28-29  
   Tower of Hanoi, 23-24  
 research areas, 18  
 semantic networks, 1012  
 temporal reasoning, 870
- Automatic reasoning, rule-based systems, 965
- Automatic test equipment (ATE), military applications, 610
- Automatic theorem provers, 537, 1115-1122
- Automaticity, cognitive psychology, 117
- Automation:  
 fixed, 35  
 flexible, computer-integrated manufacturing, 171  
 industrial, 35-39  
 joblessness and, 1054  
 office systems, 680-683  
 self-replication, 1011  
 teleoperators, 1100  
*see also Industrial automation*
- Automatix Incorporated, 933  
 vision system, robot sensing, 1033
- Autonomous decision making, industrial robots, 950
- Autonomous vehicles, 39-44  
 applications, 44  
 control and position location functions, 40  
 implementation, 44  
 learning, 43-44  
 limits, 496  
 military applications, 604, 1056  
 obstacle avoidance, 42-43  
   avoidance control, 42-43  
   detection and location, 42  
 perception, 40-41  
   landmark recognition, 41  
   obstacle detection, 40  
   road detection, 40-41  
   sensor mapping, 41  
 position location, 41-42  
   dead reckoning, 42  
   map matching, 42  
   route planning, 43  
   map transforms, 43  
   route search, 43  
 stereo vision, 1083-1084  
 vehicle control, 41
- Autonymy, self-reference, 1008
- AUTOPASS language, industrial robots, 951
- Autopoiesis, cybernetics and, 226
- Avoidance control, autonomous vehicles, 42-43
- Avoidance goals, 757
- Awit program, computer chess, 168
- Axiomatic systems  
 modal logic, 618  
 propositional logic, 561-562
- Ayer, 282
- B\* algorithm, branch-and-bound search, 1003
- Babbage, Charles, 9
- BABEL system, script applications, 984
- Background knowledge rules, clustering, 110
- Back-propagation learning algorithm, demons, 232
- Backtracking, 46-48  
 advantages and disadvantages, 46  
 AND/OR graphs, 8  
 augmented transition networks, 667  
   grammar deviations, 672  
 belief revision:  
   chronological, 58  
   MBR and TMS, 61  
   planning, 757  
 chronological vs. dependency-directed, 47-48  
 constraint propagation, 46  
 constraint satisfaction, 206-209  
 control structure, virtual machine generalization, 213  
 coroutines, 219  
   implementation, 220-221  
   nonforgetful, 221  
 dependency-directed, 47-48  
   belief revision, 58-59  
   constraint satisfaction, 207  
   default reasoning, 841  
   frame theory, 304  
   logic programming, 554  
   planning, 757  
   rediscovering contradictions, 47-48  
 depth-first search, 1005  
 intelligent, 46-47  
 knowledge representation, 884-885  
   PROLOG system, 887  
 limits of, 500-501  
 logic programming, 546-547  
 machine learning, 469-470  
 music in AI, 638  
 parsing, 142  
 pattern matching, 718  
 planning and operation, 752-753, 757  
 problem formulation, 46  
 problem reduction, 764  
 programming environment, 792  
 programming language application, 47  
 reordering variables, 46  
*see also Dependency-directed backtracking*
- Backward chaining, 48  
 image understanding, 398

- knowledge representation, 887
- knowledge system techniques, 289–290
- medical advice systems, 585, 587
- military applications, 608
- path planning and obstacle avoidance, 709–710
- pattern matching, 719
- planning actions, 749
  - inhibiting, 754
- processing, bottom-up and top-down, 780
- reasoning, 824
- rule-based systems, 968
- see also* Node expansion
- BACON system
  - clustering, 110
  - concept learning, 187
  - data-driven, 190
  - creativity, 224
- BACON.4, quantitative laws for discovery, 484–485
- Bacon/Roger, 282
- Bajcsy, R., 633–638
- Ballard, B., 133–146
- Banerji, R. B., 312–318, 614–617
- Bar-Hillel, Y., 567
  - machine-assisted human translation, 133
- Barcan formula, modal logic, 618–619
- Barnard, Stephan T., 1083–1089
- Barrow, H., 576–577
- Barstow, D., 785–788
- Bartlett, F. C., schema notion, 302
- Base grammar, 355–356
- BASEBALL system, 48
  - computational linguistics, 134–135
  - human-computer interaction, 384
- BASIC instructional program (BIP), 154–155
- Basic PDP-1 LISP, 522
- BASIC programs, legal analysis, 460
- Basic transition network, ATN parsing, 693
- Bateman, J. A., hermeneutics, 372
- Bates, M., 655–659
  - ATNs, 330
- Bates parser, 330
- Bateson, Gregory, 302
- BATTLE system, 608
- Bayes' theorem, medical advice systems, 586
- Bayesian decision methods, 48–56
  - basic formulation, 48–50
  - belief propagation in networks, 54–55
  - Boltzmann, 81
  - commonsense reasoning, 834
  - decision theory, 229–230
  - default reasoning, 741–742, 844–845
  - evidence pooling, 50
  - image understanding, 399–400
  - inductive inference, 416
  - inference, 419
  - inversion formula, 49
  - knowledge representation, 889
  - knowledge system techniques, 289
  - medical advice systems, 586
  - multihypothesis variables, 50–51
  - multiple causes and "explaining away," 53–54
  - networks, 54
  - pattern recognition, 720–722
    - optimization, 726–727
  - predicting future events, 52–53
  - prospective and retrospective supports, 49–50
  - rational decisions and quality guarantees, 55–56
  - uncertain evidence (cascaded inference), 51–52
  - uncertainty, 854–856
  - virtual (intangible) evidence, 52
  - Viterbi algorithm, 1161
- BB1:
  - blackboard architecture, 73, 78
  - knowledge engineering, 297
  - rule-based systems, 971
- BBN LISP, 522
- B. Curry package, AI in chemistry, 95
- Beacon systems, mobile robots, 958
- Beam models, image understanding, 397–398
- Beam search, 56–57, 998
  - best-first search, 999
  - CANDIDATE.STATES, 57
  - CURRENT.STATES, 57
  - image understanding, 397–398
  - machine learning, 467
    - Aq algorithm, 471
- Bebe program, computer chess, 160
  - search and knowledge errors, 169
- Behaviorism:
  - cognitive modeling, 111–112
  - cognitive psychology, 115–118
  - episodic memory, 275
  - philosophy of mind, 737–738
  - Turing test, 1127
- Being and Time*, 364, 731
- BEINGS system, distributed problem solving, 246
- Bejczy, A. K., 1100–1101
- Belief propagation, Bayesian networks, 54–55
- Belief revision, 58–62
  - applications, 61–62
  - artificial intelligence, 58–59
  - belief space, 60
  - belief systems and, 63
  - circular proof, 59–60
  - commonsense reasoning, 834
  - context, 60
  - default reasoning, 841
  - disbelieving process, 59
  - explicit concern, 58
  - foundation concerns, 60
  - justification-based *vs.* assumption-based systems, 60–61
  - origin set (OS), 60
  - origin tag, 60
  - restriction set (RS), 60
- Belief selection, default reasoning, 841
- Belief systems, 63–71
  - artificial, 70
  - circumscription, 100
  - compatibility, 71
  - constraint satisfaction, 207
  - de re* and *de dicto* theories, 64–66
  - discourse understanding, 240
  - distributed problem solving, 249
  - emotion modeling, 274
  - epistemic logic, 64–65
  - epistemology, 280–281, 285
  - extensional and intensional objects, 63
  - heuristic theories, 66–68
    - Moore, 66
    - psychological, 68–70
      - Abelson, 70
      - Colby, 69–70
      - user models, 70
      - Wilks and Bien, 69
    - subsystems, 67
- ICAI, 154, 158
- image understanding, 393
- indirect speech acts, 1064
- knowledge and, 63
- machine translation, 565
- nested beliefs, 69–70
- PARRY system, 687
- philosophical background, 63–64
- propositional attitudes, 837–838
- quantification, 63
- referential opacity, 63–64
- semantic memory, 593–594
- social issues of AI, 1055–1056
- survey of theories and systems, 64–66
  - belief spaces, 65
  - intensional theories, 65–66
- Belief theory, plausible reasoning, 857–858
- Believed propositions, 58
- Belle chess-playing system, 73, 160
  - end game play, 167–168
- Bellman's dynamic-programming algorithm, mobile robots, 960
- Berkeley (Bishop), 282
- Berliner, Hans, 160, 380
- Berwick, R., 353–361
- Best-first search, 997–999
  - problem solving, 773
  - related strategies, 999
- Beta structures, frame theory, 302
- Biblical exegesis, hermeneutics, 363
- Bidirectional heuristic front-to-front algorithm (BHFFA), 1000
- Bidirectional reflectance distribution function (BRDF), shape analysis, 1041–1042
- Bidirectional search, 996, 1000
  - problem solving, 773
  - retargeting, 1000
  - story analysis, 1091
  - strategies, 1000
- Biermann, A. W., automatic program synthesis, 28
- Bigelow, J., 226
- Bignum data structure in LISP, 518
- Binary images, shape analysis, 1044
- Binary relation:
  - case grammar, 338
  - propositional attitude, 837–838
- Binary resolution, 892–901
  - clause, 893–894
    - empty clause, 897
    - language, 895
  - current applications, 900
  - formal treatment, 895–898
  - inference:
    - factoring, 897
    - hyperresolution, 897
    - most general common instance (MGCI), 896
    - most general unifier (MGU), 896
    - paramodulation, 898

- UR resolution, 897–898
- informal treatment, 894–895
- proof finding, 895–896
- reasoning programs, 900
- representation, 895
- Skolem functions, 895
- strategy, 898–900
  - demodulation, 899
  - set of support, 899
  - subsumption, 899
  - unit preference, 898–899
  - weighting, 899
- Binary template matching, 577
- Binary tree:
  - data structures, 547
  - NON-VON computers, 678
- Binding and connectionism, 202–203
- Binet, A., 432
- Binford, T. O., 321–323
- Binocular procedures, motion analysis, 628
- Binocular stereo, shape from shading, 1044
- Biochemistry, brain and, 489
- Biological sciences, image understanding system, 407
- Biological systems, noisy data, 473
- Biomedical technology:
  - early vision, 1131
  - stereo vision, 1083
- BIRD frame language, 306
- Bisiani, R., 56–57
- Black boxes
  - built-in test systems (BITS), 610–611
  - cognitive science, 123
- Black, H., 299
- Black, J., 306
- Blackboard systems, 73–79
  - characteristic behavior, 74
  - cognitive psychology, 116, 118
  - control structures, 213
  - defining features and characteristics, 73–74
  - effective scheduling, 79
  - image understanding systems, 397–398
  - knowledge engineering, 297
  - knowledge representation, 888
  - military applications, 608
  - mobile robots, 960–961
  - motivating objectives, 73
  - objectives approach, 74–75
  - object-oriented language, 455
  - parallel computing, 79
  - path planning and obstacle avoidance, 715
  - planning, 755
  - question answering, 819
  - speech understanding, 1081
  - system-building environments, 77–79
    - AGE, 77–78
- Bledsoe's program, binary resolution, 900
- Blindness concept, 372
- BLISS system:
  - coroutines, 222
  - prostheses, 799
- Blob detection, dot-pattern analysis, 254–255
- Block design, intelligence and, 432
- Block, N., 436
- Blocks world image, 362
  - default reasoning, 843–844
- early vision, 1132, 1142
- industrial robots, 950
- object-oriented programming, 453
- path planning and obstacle avoidance, 708
- planning, 748
  - operation, 750–753
- WHISPER program, 867–869
- BNF (Backus-Naur form) grammar, 688
- Board description, computer chess, 165–166
- Bolt, Beranek & Newman (BBN) LISP, 522
- Bolt, R., 384
- Boltzmann machine, 80–81
  - constraint satisfaction, 210
- demons, 232
- Bonissone, P., 854–862
- Book learning, checkers-playing programs, 91–92
- Books on artificial intelligence, 531–532
- Boolean constraint propagation, Waltz filtering, 1162
- Boolean satisfiability, constraint satisfaction, 209
- Booth, A. D., 133
- Bootstrapping, hermeneutics, 363–365
- Boring, C., 432
- BORIS system, 81
  - discourse understanding, 240–241
  - expectation-driven parsing, 700
  - intelligent agents, 837
  - POLITICS and, 757
  - scripts, 981
  - integrated in-depth parsing, 988–989
  - problems, 985
  - story understanding, 1097–1098
- Bossie realization component, 651
- Bower, G., 306
- Brachman, R., 308, 441
- Bracket abstraction, 442
- Braille reading prostheses, 800
- Brain (human):
  - connectionism, 200
  - developmental mechanisms, 491
  - limits of quantitative theory of computational complexity, 491–492
  - physical activity of, 488–489
  - programs selected, 490–491
  - quantitative estimates of computing power, 489–490
- Brain damage and development of intelligence, 437
- Branch-and-bound search, 1000–1004
  - abstract procedures, 1001
  - algorithms, 1003
  - best-first, 1001–1002
  - complexity, 1003
  - definitions, 1001
  - depth-first, 1002
  - dynamic programming, 1003–1004
  - pruning, 1001–1003
- Branching factor, 81–82
  - alpha-beta pruning, 5
  - checkers programs, 88
- Bratko-Kopec test, computer chess, 165
- Bratko, I., 315
- Breadth-first search, 995–996
  - agenda-based systems, 3
  - best-first search, 999
  - logic programming, 547
- medical advice systems, 585
- problem reduction, uninformed search, 765
- problem solving, 773
- Breakdown concept, 372
- Brentano, Franz, 282–283
  - phenomenology, 731
- Bridge, mathematical formulation, 312
- Briscoe, E. J., 1076–1082
- British Museum algorithm, law applications, 457
- Brouwer, L. E. J., 283
- Brown, C., 382–383
- Brown, J. S., 304
- Bruce, B. C., 233–244, 339
- Brute-force search method, game-playing, 848
- Bruynooghe, M., 46–47, 219–223
- Buchanan, B. G., 233, 457–458
- BUGGY program, 155
- Built-in test (BIT) system, 610–611
- Bureau of Labor Statistics, AI and joblessness, 1054
- Bureaucracies and office automation, 682–683
- Burglary hypothesis, Bayesian decision methods, 49–50
- Burt, C., 434
- Business
  - AI in, 533
  - legal analysis programs, 458–459
- Butterfly machine, symbolic computation, 216
- CA (conceptual analyzer):
  - expectation-driven parsing, 698–699
- CADAM system (computer-augmented design and manufacturing), 151
- CAD/CAM systems:
  - equipment testing, 610
  - hardware requirements, 151
  - principal applications, 151–152
  - see also* Computer-aided design and computer-aided manufacturing; Computer-integrated manufacturing
- CADR computer system, 176–177, 529
  - computer architecture, 216
- CADUCEUS system, 82, 440
  - causal reasoning, 828
  - development of, 587–588
- Calculus:
  - lambda, 441–442
  - qualitative physics, 811
  - uncertainty, 861
- Calendar systems and office automation, 681
- Call-by-value passing convention, 513
- Cambridge Language Research Unit, 569
- Cambridge Polish, LISP development, 520
- Camera modeling:
  - stereo vision, 1084–1085
  - relative model, 1085
  - template matching, 576
- Canny's edge detection algorithm, 263–266
- Carbonell, Jaime G., 464–485, 660–675
- Carbonell, J. R.
  - intelligent computer-aided instruction, 154–159
- SCHOLAR, 980



- Cardozo, B., 457
- Carnegie-Mellon University (CMU) vehicles, 40
- Carotid pulse waves, 727
- CARPS system for computational linguistics, 135, 140
- Carrier Sense Multiple Access-Collision Detection (CSMA-CD), 953
- Carroll, J., 436
- Cartesian coordinates, wire-frame representation, 891
- Cartesian intuition, 737
- Cascaded augmented transition network (CATN), 332
- Case frame:
- case structure, 334
  - deep structure, 231
  - ellipsis resolution, 673-674
  - frame theory, 302
  - instantiation:
    - conceptual dependency, 669-670
    - natural-image understanding, 668-671
  - parsing, 670-671
  - required, optional, and forbidden cases, 669
- KL-ONE system, 335
- properties, 335
- selection restrictions, 334
- word-expert parsing, 705
- Case grammar, 333-338
- case systems, 335-338
  - computational linguistics, 136
  - deep case theory:
    - grammatical explanation, 333-334
    - meaning representation, 334-335
  - frame theory, 308
  - surface cases, 333
  - systems, Kasus functions, 336
- Case markers, deep structure, 336
- Case structure, 336
- Case system, 333
- Grimes system, 336-337
- Case theory:
- frame theory, ISA hierarchy, 308
  - transformational grammar, 358
- Case-based reasoning, 477-478
- memory organization packets, 591
- Case, J., 875-876, 1123-1125
- CASE system for chemistry in AI, 95
- CASREP system, 612-613
- causal relationships, 828
  - clinical use, 590
  - development of, 588-589
  - domain knowledge, 252
  - medical advice systems, 585
- Casual conversation discourse understanding, 243
- CAT scans, pattern recognition, 727
- Cattell, R., 434
- Causal modeling, medical advice system, 584-585
- Causal net representation, medical advice systems, 588
- Causal reasoning, 827-832
- Bayesian decision methods, 49
  - causal-link models, 827-828
  - cognitive mental models, 831-832
  - defining causality, 830-831
  - symbolic simulation, 828-830
- Causality:
- physics, 747
  - speech acts, 1063
  - story analysis, 1094
  - temporal reasoning, 872
- CCD chips, color imaging, 124-126
- C-cells, path planning, 712
- Celce's deep case grammar system, 336
- Cell automata, self-replication, 1010-1011
- Cell migration in brain, 490-491
- CENTAUR system, 4
- Certainty in artificial intelligence, 14
- Buchanan-Shortlife modification, 857
  - confirmation theory, 856-857
  - epistemology, 281
  - Heckerman definition, 857
  - hypothesis ranking, 393
  - knowledge system techniques, 289
  - medical advice systems, 586-587
  - S.I expert system, 291
- CGOL, 510-511
- Chaining, *see* Forward and Backward Chaining
- Chaining
- Chaos chess program, 160
- end game play, 167
  - search and knowledge errors, 169
  - selective search, 168
- Character recognition, 82-87
- applications, 86-87
  - contextual processing, 85-86
  - binary *n*-grams, 85-86
  - DVA, 86
  - Trie, 86
- feature analysis, 83
- F description, 83-84
  - functional, skeletal and physical levels, 84
  - isolated characters, 84
  - knowledge source organization, 84-85
- pattern recognition, 728
- reading prostheses, 800
- template matching, 83, 576-577
- Characteristic dimension in texture analysis, 1111
- Charniak, E., 304
- MICRO-PLANNER, 603-604
- Chart parsing, 691-692
- augmented transition networks, 667
  - speech understanding, 1080
- Chattering, robot control, 918
- Checkers-playing programs, 88-93
- alpha-beta pruning, 4-6
  - assigning credit and blame, 474-475
  - game trees, 319
  - heuristics and phase table method, 92
  - history, 9
  - human expertise, 733
  - learning, 90-92
  - generalization, 91-92
  - linear polynomial approach, 91
  - nodes for, 317
  - rote learning, 90
  - signature tables, 91-92
- parallel trees, 92-93
- representation, 89-90
- Samuel's work, 89
- search, 90
- dynamic ordering, 90
  - fixed ordering, 90
  - forward pruning, 90
  - strategy efficiency, 312
- CHECKERS program, concept learning, 186
- Chemical synthesis
- limits of, 497
  - tree, 94
- CHEMICS system, 95
- Chemistry, artificial intelligence, 93-97
- agricultural chemicals, 96
  - computer-aided education, 96
  - computer algebra applications, 96
  - experimental design, 96
  - future applications, 96-97
  - graphics, 94
  - instrumentation improvement, 95-96
  - macromolecular structure determination, 96
  - natural- and chemical-language applications, 93-94
  - reaction synthesis, 94
  - structure elucidation, 94-95
  - candidate structures, 95
  - enumeration, 95
  - fragment determination, 94
  - infrared spectroscopy, 94
  - internal consistency checks, 95
  - mass spectroscopy, 94
  - nuclear magnetic resonance, 94-95
  - x-ray powder diffraction, 95
  - water chemistry in steam power plants, 96
- Chess:
- alpha-beta pruning, 4-6
  - cognitive modeling, 113
  - computer methods, 159-169
  - heuristics, 377
  - game tree, 319
  - horizon effect, 380-381
  - strategy efficiency, 312
  - see also* Computer chess
- Chess programs:
- 4.0, 160
  - 4.4, 168
  - 4.5, 99
  - 4.6, 160
  - 4.9, 168
- CHI programming assistant, 788
- automatic programming, 33
- CHILD system of language acquisition, 145-146
- Chinese room puzzle, 738-739
- social issues of AI, 1053
  - Turing test, 1129
- Chomsky, Noam:
- aspects model, 692
  - control structure, 214
  - deep structure, 325
  - language acquisition, 443-444
  - langue-parole dichotomy, 503-504
  - normal form grammar, 689
  - perceptrons, 730
  - transformational grammar, 134, 353-354
  - X-bar theory, 479-480
- Chromatography, chemical instrumentation, 95-96

- Chronicles, temporal reasoning, 871
- Chunks, rule-based systems, 967
- Church's thesis, 99–100
  - binary resolution, 897
  - social issues of AI, 1050–1051
  - Turing machines, 1125
- Church, A., 441
- Church-Gödel theorem, 491–492
- Church-Rosser theorem, 442
- Chutes and ladders game, 269
- CIE commission, color imaging, 125–126
- CIELUV system, 127
- CIP programming assistant, 788
- Cipra, R., 571–575
- Circle detection, Hough transform, 382–383
- Circuit chips, integrated, 678
- Circuit debugging programs, temporal reasoning, 870
- Circumscription, 100–102
  - applications and related work, 102
  - completeness results, 102
  - default reasoning, 843
  - domain, 102
  - efficiency, 102
  - formula, 101
  - generalization, 101
  - knowledge representation, 889
  - minimal models, 101
  - nonmonotonic reasoning, 852
  - predicate, 100–102
  - soundness, 102
  - theoretical basis for, 101–102
  - variable, 101
- Citrenbaum, R. L., 315
- CK-LOG (Calculus for Knowledge processing in Logic) system, 609
- CKY parsing, 691–692
  - algorithm implementation, 694
- C language:
  - connection machines, 178–179
  - industrial robots, 950
  - LISP interpretation, 518
  - semantic networks implementation, 1022
- C\* language, 178–179
- Classification:
  - pattern recognition, 720
  - multistage, 723–724
  - problem solving, 768
- Classifier algorithm, inheritance hierarchy, 427
- Clauses, logic programming, 545–546
  - addition or deletion, 548
  - theorem proving, 1116
  - deletion, 1119
  - resolution, 1118
- CLISP system, 510
- InterLisp, 523
- Closed-loop systems:
  - computer-interpreted manufacturing, 172
  - robot arms, 904–906
  - stability, 904–905
- Closed systems in physics, 836
  - reasoning, 100
- Closed world assumptions (CWA):
  - Horn-clause forms, 549
  - inference, 419
  - knowledge representation, 889
  - nonmonotonic reasoning, 849
- CLOUT system, semantic grammar, 350
- Clue words, discourse understanding, 237
- CLUSTER/2 program, 107–108, 483
- CLUSTER/S program, 107–108
- Clustering, 103–110
  - classical view, 103
  - vs. conceptual view, 103–105
  - classification, 105
  - color space, 128–129
  - conceptual, 107–110, 187
  - machine learning, 481–483
  - methods, 483
  - microcomputers, 108
  - trains, 108–109
  - dot-pattern analysis, 255–256
  - military applications, 607–608
  - multidimensional, 105
  - one-dimensional, 105
  - pattern recognition, 720–721, 723
    - medical advice systems, 727
    - science, 728
  - similarity-based, 103–107
    - agglomerative techniques, 105
    - direct techniques, 105
      - center adjustment method, 105
      - k-means, 105
    - divisive techniques, 105
    - two-dimensional, 105
- CM LISP language, 178–179
- CMU Terragator robot, 950
  - navigation, 953
- COBWEB system, 483
- CODASYL database, 138
- Code analysis, information processing, 436
- Coding modules, natural-language automatic programming, 31–32
- Coding sheet project, character recognition, 84–85
- Coelho, H., 339–342
- Cognitive economy, 113
- Cognitive inpenetrability, control structure, 214
- Cognitive mapping, route finding, 866–867
- Cognitive modeling, 111–114
  - artificial intelligence and, 114
  - basic approaches, 112
  - computational linguistics, 145
  - concept learning, 185
  - emotion modeling, 274
  - evaluation, 112
    - behavioral data, 112
    - empirical quality, 112
    - theoretical quality, 112
  - language acquisition, 443–450
  - language comprehension, 113–114
  - learning, 113
  - memory organization and process, 113
  - perception, 112–113
  - problem solving and reasoning, 114
  - purposes, 111–112
- Cognitive process:
  - early vision, 1132
  - hermeneutics, 363
  - law applications, 461
  - meta-knowledge and meta-reasoning, 598–599
  - prostheses, 800
  - word-expert parsing, 701
- Cognitive psychology, 115–118
  - artificial intelligence and, 116
- cognitive science and, 120–121
- communicative action, 367
- concept learning, 185
- education applications, 267
- emotion modeling, 274
- future prospects, 118
- history and scope, 115–118
- human-computer interfaces, 387
- modeling, 111–114
- natural-language processing, 661
- research areas, 116–118
  - attention, 117
  - language, 118
  - memory, 117–118
  - perception, 116–118
  - thinking, 118
- semantic memory, 593
- social issues of AI, 1059
- Cognitive Science*, 121
- Cognitive science, 120–123
  - analogue representation, 881
  - architectures, 120
  - artificial intelligence, 120–123
  - causal reasoning, 831–832
  - characteristics, 122–123
  - concept learning, 185
  - demons, 232
  - education applications of AI, 270
  - empirical fit, 120
  - expertise and qualitative reasoning, 122
  - hermeneutics, 369, 372
  - human performance models, 122
  - intelligence and information processing, 435–436
  - intelligent computer-aided instruction (ICAI), 158
  - knowledge representation, 882–883
  - language, 121
  - learning, 122
  - literature for AI, 531–532
  - representational metapostulate, 123
  - research problems, 121–122
  - social issues of AI, 1059
  - sufficiency condition, 122
  - Turing machines, 1125
  - vision, 121–122
- Cohen, H., 8–9
  - speech act theory, 68
- Coherence relations, 281
  - discourse understanding, 235–236
- Collins, A., 204, 304
- Collision avoidance:
  - collision-detection algorithm, 711
  - industrial robots, 952–953
- Colmerauer, A.:
  - definite-clause grammar, 340–342
  - tree grammar, 339
- Color:
  - artificial intelligence limits, 493
  - early vision, 1144
  - ratios, feature extraction, 300–301
  - space, 126–127
  - vision, 124–130
    - color spaces and transformations, 126
    - imaging, 124–126
    - perceptual variable, 130
    - physical or perceptual quantity, 130
    - quantitative analysis, 130
    - statistical quantity, 128–130

- Combat resource allocation, 608–609
- Combinations:
  - LISP system, 514
  - simple evaluation, 516
- Combinatorial explosion problem, multisensor recognition, 636
- Combinatory terms, lambda calculus, 442
- Command, language:
  - communications and intelligence
  - language user interfaces, 386
- Command logic, 537
- Common LISP, 508
  - definition, 526
  - DEFSTRUCT, 514–515
  - history, 526
  - NIL symbol, 512
  - numbers, 518
  - special variables, 511
- Commonsense reasoning, 3
  - causal reasoning, 829
  - circumscription, 100
  - default reasoning and, 844
  - discourse understanding, 239–240
  - domain models, 834–835
    - quantities, 834–835
  - emotion modeling, 274
  - frame theory, 302–303, 834
  - hermeneutics, 368
  - inference, 419, 833–834
  - intelligent agents, 837
  - interpersonal interactions, 838
  - methodology, 833
  - naive physics and, 746–747
  - ontology, 833
  - physical, 836–837
  - plausible inferences, 834
  - propositional attitudes, 837–838
  - social issues of AI, 1051
  - story analysis, 1091–1092
  - time, 835–836
  - word-expert parsing, 702
  - see also* Reasoning, commonsense
- Communication:
  - capabilities, knowledge system techniques, 290
  - discourse understanding, media selected, 243–244
  - image representation, 1136
  - language and, 838
  - natural-language understanding, 660
  - object-oriented programming, 452–453
  - office automation, 682
  - receiver-sender, 221
  - robotics, 953
  - speech act theory, 1062–1065
  - systematic distortion, 366
  - top-down, AI social issues, 1055
- Communicative action theory, 367
- Comparison, image understanding, 392
- Compatible Time Sharing System (CTSS), 522
- Competence:
  - control structure, 214
  - defined, 503–504
  - hermeneutics and, 366
  - in linguistics, 503
- Competitive learning, demons, 232
- Compilers:
  - coroutines, 221
- LISP:
  - separate, 526–527
  - 1.5, 521
- Completeness, 131–132
  - binary resolution, 895
  - circumscription:
    - negative results, 102
    - positive results, 102
  - logic and, 536–537
  - predicate logic, 543
  - proof procedures, 132
  - theorem proving, 1116
  - uncertainty and, 854
- Composition, musical, AI and, 638–641
- Compositionality and semantics, 1025
- Compounding and morphology, 619
- Comprehension, intelligence and, 432
- Computational architecture, Bayesian networks, 55
- Computational geometry, computer-aided design, 152
- Computational hardware, autonomous vehicles, 44
- Computational linguistics, 133–146
  - application areas, 137–138
  - database interface, 137–138
  - augmented transition network parsing, 693–694
  - automatic programming, 140–141
  - broadening interests (1960–1970), 134–135
    - consultation, 135–136
    - problem solving, 135
    - question-answering systems, 134–135
  - case frame instantiation, 668–669
  - cognitive modeling, 145
  - computer-aided instruction, 139–140
  - current trends, 141
  - discourse understanding, 143–144
  - domain-independent implementations, 141
  - early work (1950–1965), 133–134
  - formalism developments (1965–1970), 136–137
  - generalized phrase structure grammar, 343
  - grammatical formalisms, 142–143
  - information retrieval, 134
  - language acquisition, 145–146
  - letter-to-sound rules, 1073–1074
  - limits of, 495
  - LUNAR, 137
  - machine translation, 133–134
    - re-emergence, 141
  - machine translation, 564–565
  - natural language processing:
    - commercialization, 141
    - generation, 643
  - office automation, 140
  - parsing, 141–143, 695
  - scientific text processing, 141
  - semantic networks, 1012
  - SHRDLU, 137
  - speech understanding, 146
  - text generation, 144–145
  - theoretical issues, 141–146
  - transformational grammar, 134
  - ungrammaticality, 143
- Computational logic, machine translation, 565
- Computational stereo, 1083–1089
- Computational theory of mind, 738
  - anticomputationalist critique, 738–739
  - replies to anticomputationalist critique, 739
- Computational vision, 1131
  - cognitive science, 121
  - parallelism, 1139
  - tasks, tools, techniques, 1141
- Computed torque algorithm, robot-arm control, 917
- Computer-aided design (CAD), 151–153
  - AI applications, 152–153
  - computer-integrated manufacturing, 172
  - engineering, 36–37
  - faces, 934
  - industrial robot control systems, 949–950
  - natural-language interfaces, 656
  - parameterized, 153
  - robotics, 941
  - shape analysis, 1046
  - wire-frame representation, 892
- Computer-aided instruction (CAI):
  - chemistry, 96
  - computational linguistics, 139–140
  - intelligent, 154–159
  - SCHOLAR system, 980
  - see also* Intelligent computer-aided instruction (ICAI)
- Computer-aided manufacturing (CAM), 151, 892
  - robotics, 941
- Computer architecture:
  - control structure, 215–216
    - symbolic computation, 216–217
  - DADO, 227–228
  - mobile robots, 960–961
  - new generation, 678
  - NON-VON computers, 678
- Computer chess methods, 159–169
  - alpha-beta algorithm, 161–162
  - aspiration search, 162
  - end game play, 167
  - forward pruning, 162–163
  - from-to square combination, 163
  - future programs, 169
  - hardware advances, 166
  - heuristics, 377
  - history, 9
  - horizon effect, 163–164, 380–381
  - human expertise, 733–735
  - implications, 160
  - landmarks, 159–160
  - memory tables, 167–168
  - minimal game tree, 162
  - minimal window search, 162
  - move reordering mechanisms, 163
  - phenomenology, 741
  - progressive and iterative deepening, 164
  - quiescence search, 163
  - reasoning, 848
  - search and knowledge errors, 169
  - search methods, 994
  - selective search, 168–169
  - semantic networks, 1012
  - social issues of AI, 1058
  - software advances, 166–167

- strengths and weaknesses, 165–169
- terminology, 160–161
- transposition and refutation tables, 164–165
- Computer-integrated manufacturing (CIM), 171–173
  - artificial intelligence and, 172–173
  - robotics, 953–954
  - shape analysis, 1046
- Computer models of language processing, transformational grammar, 353
- Computer Professionals for Social Responsibility, 1056
- Computer programming, defined, 18
- Computer science, social issues of AI, 1050
- Computer systems, 174–180
  - AI-based
    - vs. conventional programs, 175–176
    - examples, 176–179
    - languages, 174–175
  - connection machine, 177–179
  - control strategy, 175
  - dynamic data typing, 175–176
  - evolutionary versus revolutionary architectures, 180
  - human interface, 175
  - instruction set level, 176
  - memory management/processor scheduling, 176
  - new concepts, 180
  - orientation, 180
  - storage management, 175
- Computer understanding:
  - capacity, 373
  - conceptual dependency, 198
- Computer vision:
  - analogue representation, 881
  - heuristics, 377
  - Hough transform, 382–383
  - human-computer interaction, 384
  - limits, 496
  - motion analysis, 620
  - NON-VON system, 678
  - optical flow, 684
  - texture analysis, 1102
  - Waltz filtering, 1162
- Computers and Thought*, 48
- Computers in education, 181–185
  - alienated vs. syntonic learning, 184
  - computer culture, 182
  - curriculum transformation, 183
  - density, 181
  - educational technology, 183–184
  - ideology, 181
  - instruction vs. development, 181
  - instruction manuals, videodisks and programs, 182
  - learner control of, 183–184
  - locus of learning, 182–183
  - social interaction, 184–185
  - structured, metacognitive and developmental perspectives, 181–182
- Comtex Scientific Corporation, 530
- Concept acquisition, 122
- Concept description, 466–467, 475
- Concept discovery, problem solving, 778
- Concept learning, 15, 185–192, 464–465
  - analogous, 186
  - analytic approaches, 471–473
  - classical view, 192
  - clustering and, 107
  - data-driven, 190
  - deductive, 186
  - defined, 185
  - exemplar view, 192
  - extended notions, 191–192
  - image understanding, 391
  - incremental specialization, 190–191
  - inductive, 186–187, 190
    - hypothesis generation, 187–188
    - rules, 189
  - inference, 186–187
  - instance space vs. description space, 189–190
  - instructive, 186
  - knowledge implantation, 186
  - machine learning, search, 466–467
  - mixed methods, 191
  - model-driven methods, 190–191
  - observation and discovery, 186–187
  - preference criterion, 187–188
  - premise statements, 188
  - probabilistic view, 192
  - problem solving, 777
  - typicality, 192
  - validation, 191
- CONCEPT variable, expectation-driven parsing, 696
- Conceptual analysis, word-expert parsing, 702
- Conceptual cases, 337
- Conceptual clustering, 103–105
  - cohesiveness, 104–105
  - machine learning, 481–483
- Conceptual coverage, menu-based natural language, 594
- Conceptual dependency, 194–199, 371
  - ATRANS, 197–198
  - ATRANS-CONSEQS, 198
  - ATTEND, 198
  - BORIS system, 988–989
  - case frame instantiation, 669–670
  - computational linguistics, 136
  - computer understanding, 198
  - conceptual nominals, 195
  - economy, 197
  - expectation-driven parsing, 696–697
  - EXPEL, 197
  - FRUMP system, 988
  - GRASP, 197
  - hermeneutics, 371
  - human actions, 837
  - inferences, 198
  - INGEST, 197
  - language generation, 198–199
  - limits of artificial intelligence, 495–496
  - machine translation, 570
  - MBUILD, 198
  - meaning similarity, 197
  - MOVE, 197
  - natural language generation, 649
  - need for conceptual primitives, 196–197
  - parsing, 199
  - primitives, 759
  - PROPEL, 197
  - PTRANS, 197–198
  - rules, 195–196
    - rule 1, 195
    - rule 2, 195
    - rule 3, 195
    - rule 4, 195–196
    - rule 5, 196
    - rule 6, 196
- SAM system, 986–987
- script, 981
- semantic networks, 1012, 1025
  - graphs, 1019
  - natural language generation, 646–647
- SPEAK, 198
- static forms, 198
- story analysis, 1091, 1093
  - memory organization packets (MOPs), 1097
- structure, 194–195
- Conceptual factoring, ATNs, 325
- Conceptual graphs, semantics, 1025
- Conceptual retrieval, legal applications, 460
- Conclusions in logic, 536
- Concurrency, distributed problem solving, 249
- Concurrent languages and coroutines, 222
- Concurrent PROLOG, 556–557
  - computer architecture, 216
  - coroutine, 222
- Cones:
  - object models, 633
  - sensor mapping, 41
  - see also Generalized cone
- Conference literature on artificial intelligence, 531, 533
- Configuration space:
  - manipulation, 573–575
  - path planning and obstacle avoidance, 710–711
    - mobile robot applications, 711
    - object expansion, 710–711
    - representation as spheres, 711
    - robot arm, 711
- Confirmation theory:
  - certainty factor, 856–857
  - uncertainty and, 854
- Conflict resolution:
  - control structures, 213
  - goal hierarchies and ordering, splicing, 756
  - knowledge representation, 887
  - military applications, 607
  - planning process, 752
- Connection graphs, theorem proving, 1120
- Connection machines, 199–200
  - computer architecture, 216
  - computer system, 174
  - hardware, 199–200
    - host machine, 199
    - router, 199
  - operations, 177–179
  - software, 200
- Connectionism, 200–205
  - applications and summary, 204–205
  - basic model, 201–202
  - demons, 232
  - distributed, 203–204
    - network construction, 203–204
    - problem solving, 246
  - localist, 202–203

- cross-talk, binding and learning, 202–203
- lateral inhibition and positive feedback, 202
- predictability and convergence, 203
- motivations, 200–201
- Connectionist architecture:
  - Boltzmann machine, 80–81
  - language learnability, 446
- Connectionist neural networks, cognitive psychology, 116
- Connectives and propositional logic, 559
- Connectors, problem reduction, 763
- CONNIVER system, 205
  - control structure, 214
  - HACKER and, 362
  - knowledge representation, 883
  - MICRO-PLANNER and, 604
  - programming environment, 792
- CONS cell, 176–177
  - computer architecture, 215–216
  - history, 520–521
  - LISP machine, 529
  - lists, 510
  - run time typing, 517–518
- Consciousness:
  - philosophy of mind, 740
  - self-reference, 1009–1010
- CONSIGN system:
  - industrial robots, 951–952
  - robot sensing, 1033–1034
- Consistency:
  - algorithms:
    - constraint satisfaction, 207–209
    - generate-and-test, 207
  - completeness, 131–132
  - Turing machine, 740
  - tests, nonmonotonic logic, 850–851
- Constituent structure, linguistic competence and performance, 506
- Constraint propagation:
  - dependency-directed backtracking, 48
  - expectation-driven parsing, 697–698
  - image understanding, 399–400
  - programming environment, 792
  - transformational grammar, 357–358
  - Waltz filtering, 1162–1165
- Constraint satisfaction, 205–210
  - applications, 209
  - arc consistency, 208–209
  - backtracking and consistency, 207–209
  - Boltzmann machine, 80
  - Boolean problems (CSP), 206–207
  - connectionism, 200
  - n*-consistency, 209
  - controlled-continuity, 1155
  - early vision, 1138–1139
    - relaxation labeling, 1140
  - education applications, 268–269
  - functional unification grammar, 651
  - heuristics, 378
  - knowledge systems, 288
  - machine learning, 470
  - path consistency, 208
  - planning, 754
    - goal hierarchies and ordering, 755
  - postal address-reading machines, 86
  - problem solving, 769, 776–777
  - qualitative physics, 808
  - REF-ARF programs, 876–877
  - relaxation algorithms for optimization problems, 209–210
  - Waltz filtering, 1165
- Constructive solid geometry models, 152–153
- Consultation:
  - computational linguistics, 135–136
  - domain knowledge, 252
- Contact image sensing, 39
  - robot sensors, 1035
- Context:
  - ellipsis, 273
  - phenomenology, 741
- Context-dependence:
  - interpretation, 238
  - logical formalism, 239
- Context-free grammar, 326–327
  - ATNs, 325–326
  - definite-clause grammar and, 339
  - k*-tails, 414
  - language acquisition, 444–445
  - limits of, 494–496
  - music applications, 641
  - parsing, BNF grammar, 688
  - phenomenology and, 733–734
  - phrase-structure grammar, 344–346
  - problem reduction, 763
  - search, inductive inference, 412
  - self-embedding symbol, 326–327
  - sentential form, 688
  - speech recognition, 1068
  - syntactic analysis, 664–665
  - transformational grammar, 355–356
- Context-sensitive grammar, 689
  - phrase-structure grammar, 344–346
  - transformational grammar, 356
- Context space theory, discourse understanding, 236–237
- Contextual meaning, word-expert parsing, 705–706
- Contingent propositions, propositional logic, 561
- Continuity, generalized cylinder representation, 322
- Continuity of interest, legal analysis programs, 458–459
- Contract net formalism, problem solving, 248
- Contraction of programming, 793
- Contradictions:
  - belief revision, 58
  - propositional logic, 561
- Control:
  - algorithms, medical systems, 585–586
  - distributed, 246
  - heuristics, discourse understanding, 242
  - knowledge, problem solving, 769
  - robotics, 902–921
- Control agreement principle (CAP), 343
- Control structures, 211–217
  - actor communications, 2
  - computer architecture, 215–216
  - symbolic computation, 216–217
  - history, 212
  - industrial robots, 949–950
- intelligent computer-aided instruction (ICAI), 157
- knowledge representation, 887
- knowledge system techniques, 289
- pattern matching, 718
- processing, bottom-up and top-down, 779
- programming language design, 212–214
  - generalization of serial virtual machines, 213–214
  - modularity philosophies, 212–213
- representation design, 214–215
  - competence and performance, 214
  - logic programming, 215
  - priority of competence research, 214
  - procedural-declarative controversy, 214–215
- robot arms:
  - feedback control, 904–909
  - frequency-domain techniques, 903
  - set-point regulation, 904
- word-expert parsing, 705
- Convergence:
  - connectionism, 202–203
  - perceptrons, demons, 232
  - sensory data, multisensor integration, 637
- Conversational implicatures, 144
- CONVERSE system, computational linguistics, 135
- Convex decomposition, path planning and obstacle avoidance, 712
  - C-cells, 712
  - maximum-area decomposition, 712
- Convolved images, feature acquisition, 1086
- Convolution values:
  - early vision, 1133
  - feature extraction, 300–301
  - template matching, zero crossings, 577
- Conway approach to game playing, 314
- COOP system:
  - database interface, 138
  - discourse understanding, 144
  - question answering, 818
- Cooperative algorithms, constraint satisfaction, 209
- Coordination-networks:
  - distributed problem-solving, 246
  - information retrieval, 420
- Corkill, D., 245–250
- CORONATION system, 307
- Coroutines, 219–223
  - applications, 221–222
    - compilers, 221
    - operating systems, 221–222
    - receiver-sender communications, 221
    - simulation, 222
  - artificial intelligence, 222–223
  - concurrent languages, 222
  - lazy evaluation, 222
  - object-oriented programming, 222–223
- control structure, 213
- explicit *vs.* implicit sequencing, 219
- implementation, 220–221
  - backtracking, 220–221
  - semantic differences, 220
  - spaghetti stack, 220
- multipass algorithms and pipelines, 220
- reactivation point, 220
- reasoning, 848

- relation with procedures, 219
- symmetric *vs.* semisymmetric, 220
- Correlation matching, stereo vision, 1087, 1089
- Correspondence, stereo vision, 930–931
- Cosmic Cube, symbolic computation, 216
- Cottrell, G., 204
- Coulomb friction in robot-control systems, 912–913
- Coverage, natural-language interface, 657–658
  - lexical coverage, 657
  - semantic coverage, 657
  - syntactic coverage, 657
- Craik, F., 117
- Cray Blitz system, 160
  - memory tables, 168
- Creativity, 223–225
  - art in AI, 8–9
  - computer-integrated manufacturing, 171–172
  - intelligence, 432
- Credibility and social issues of AI, 1058
- Credit assignment:
  - machine learning, 474
  - problem solving, 474–475
- Creeger, Mace, 528–530
  - LISP machines, 528–530
- Critique of Pure Reason*, 282
- CRITIQUE system in office automation, 140, 143
- CROSS.ARRAYRULES knowledge system, 76
- Cross-correlation, normalized, 576
- Cross-talk and connectionism, 202–203
- Crossword puzzles and constraint satisfaction, 206–207
- Cross-world identification problem, 836
- Crowley, J. L., 708–715
- Cryptarithmic problems:
  - constraint satisfaction, 209
  - problem solving, 776
- CRYALIS system and blackboard architecture, 73, 76–77
- Crystallized intelligence, 435, 437
- Crystals*, 641
- CSL program, 191
- C3I system, 604
- CUBE, LISP extension, 526
- Culicover, P., 445
- Cullingford, R., 304, 980–992
- CUP concept, 472–473
- Curet, George, 528–530
  - LISP machines, 528–530
- Curriculum and education applications of AI, 270
- Curve detection, 1146–1147
  - flow patterns, 1146–1147
  - lines *vs.* tangents, 1146
  - parallel surface contours, 1146–1147
- Curved-surface patches in model-based object recognition, 633
- CWR (contents of the word in register numbers), LISP development, 520
- Cybernetics, 225–227
  - epistemology, 226–227
  - industrial robots, 946
  - literature, 530
  - mechanisms, 226
- CYRUS:
  - memory organization packets, 592
  - script acquisition, 992
- DADO, 227–228
  - granularity issues, 228
  - parallel architecture, 227–228
- Dahl, V., 339
- Damping coefficient, robot-control systems, 906
- DARPA system, military and political issues, 1056–1057
- Dartmouth Summer Research Project on Artificial Intelligence, 519, 530
- Dasein* (being-in-the-world), 364–365, 371–372
- Database:
  - as data structure, 552
  - deductive, closed-world assumption, 419
  - interface, computational linguistics, 137–138
  - logic programming, 551–552
  - multisensor integration access, 636
  - wire-frame representation, 891
- Database management systems (DBMS):
  - INTELLECT system, 431
  - natural-language interface, 655–656
- Data-directed programming, environment, 792
- Data-driven processing, bottom-up and top-down, 781
- Data-flow, eager execution and, 555
- Data-Representation Advisor (DRA), 787
- Data structures:
  - databases as, 552
  - programming assistant, 788
  - symbolic, connection machines, 200
- Dattatreya, G. R., 720–728
- Davis, E., 832–838
- Davis, Laura, 604–613
- Davis, M., 100
- DEACON system, computational linguistics, 135
- Dead reckoning:
  - autonomous vehicle position location, 42
  - mobile robots, 958
- DeBessonnet, C. G., 457
- Debugging procedure, Shapiro synthesis algorithm, 28
- Decision-making:
  - AI and, 1054
  - autonomous, industrial robots, 950
  - computer chess methods, 160
  - predicate logic, 543
  - social issues of AI, 1055
  - thinking and, 118
- Decision support systems for natural-language interface, 655–656, 659
- Decision theory, 229–230
  - anthropomorphic robots, 945
  - Bayesian, 48–56
  - formal decision analysis, 229
  - pattern recognition, 721–723
- Decision tree:
  - ATNs, 325
  - character recognition, 83
  - construction, 469–470
  - decision theory, 229
  - pattern recognition in science, 728
  - speech-vowel recognition, 724
- Declarative knowledge, 113
- Declarative programming, 269
  - logic, 551–552
- Decoupling, linearization and poles assignment (DLPA) algorithm, 42
- DEDALUS system and pattern matching, 719
- Deductive logic, 536–537
  - argument, 536
  - automatic programming, 20
  - game playing and learning, 317
  - predicate logic, 541
  - thinking and, 118
- Deep case theory:
  - case grammar, 333, 335
  - case markers, 336
  - Celce's system, 336
- Deep structure, 134, 230–231
  - augmented transition network grammar, 325
  - case grammar, 333
  - language competence-performance dichotomy, 507
  - transformational grammar, 355
- Default reasoning, 840–846
  - belief revision, 61
  - frame theory hierarchies, 307
  - fuzzy sets and logic, 845–846
  - inheritance, 422–423
    - programming language, 430
  - knowledge representation, 885, 889
  - legal analysis program, 459
  - logic-based approaches, 842–844
    - monotonic, 844
    - non-monotonic, 842–843
  - measure-based approaches, 844–845
  - neat/scruffy debate, 537
  - uncertainty, 854–855
- Defense Advanced Research Projects Agency (DARPA), 604
  - knowledge-based systems, 607–608
- Definite-clause grammar (DCG), 142, 339–342
  - augmented transition network, 329–330
  - Colmerauer analysis, 340–342
  - extensions of, 342
  - logic grammars, 339–340
  - natural language analysis, 340
- DEFSTRUCT, 514–515
- Degrees of freedom (DOF), mobile robots, 957
- De Groot, A. D., computer chess, 159
- Dejong, G.
  - FRUMP program, 305, 312
- De Kleer, J., 47–48, 807–813
  - qualitative physics, 807–814
- Delaunay graph, dot-pattern analysis, 255
- Delegation in objected-oriented programming, 453–454
- DELTA system, 611
- Demodulation, theorem proving, 1118–1119
- Demons, 232
  - BORIS scripts, 988–989
  - DYPAR system, 700
  - if-added, 307, 429
  - inheritance hierarchies, 423, 429
  - IF-ADDED, 429



- IF-NEEDED, 429
  - slot constraints, 426
- natural-language generation, 647
- PLANNER system, 887
- reasoning, 848
- rule-based systems, 971
- word-expert parsing, 704–705
- De Morgan laws of creativity, 224
- Dempster's rule of combination, belief theory, 857–858
- Dempster-Shafer theory, 845–846
  - belief theory, 857–858
  - evidential reasoning, 858
  - image understanding, 399–400
  - uncertainty, 854
- DENDRAL system, 233
  - agenda-based system, 3–4
  - chemistry, 93
  - candidate ranking, 95
  - control structure, 214
  - domain knowledge, 251
  - knowledge engineering, 288
  - law applications, 457
  - mass spectroscopy, 94
  - medical advice, 584
  - problem solving, 776–777
    - generate-test schema, 776–777
    - representational shift, 777
- Dendrogram, clustering techniques, 105–106
- Dennett's scheme, 738
- Deontic logic, 537
  - law applications, 459
- Dependencies of propositions in belief revision, 58–59
- Dependency-directed backtracking, 46–48
- Dependency record in belief revision, 58–59
  - see also* Backtracking, dependency-directed
- Depth:
  - acquisition, 1152
  - maps, 1152
  - representation, 1153–1154
- Depth-first iterative deepening, 996
- Depth-first minimax procedure, alpha-beta pruning, 4–6
- Depth-first search, 996, 1004–1005
  - best-first search, 999
  - checkers-playing programs, 90
  - computer chess methods, 160–161
  - coroutining, 221
  - early vision and relaxation labeling, 1140
  - logic programming, 546
  - knowledge representation, 887
  - machine learning, 467
  - path planning, 710
  - problem reduction, 764
    - uninformed search, 765
  - problem solving, 773
  - recursive-descent parsing, 691
  - speech understanding, processing strategies, 1080
- Derivation:
  - analogy, 477–478
  - context-free grammar, 688
  - morphology, 619
- Derivational theory of complexity (DTC), 507
- Derivation-formation spectrum, problem solving, 771–772
- De Saussure, F., 503
- Descartes, Rene, 281
- Design, computer-aided (CAD), 151–153
  - engineering, 36–37
  - see also* Computer-aided design (CAD)
- Design tasks, programming assistants, 786
- Destructive list operations, LISP 1.5, 521
- Determinism:
  - computer-integrated manufacturing, 173
  - word-expert parsing, 702
- Deutsch, L. Peter, 522
- Developmental psychology:
  - intelligence and, 437
  - language acquisition, 118
- Device functioning, qualitative physics and, 807–808
- DEVISER system, 756–757
  - robotics planning, 941
- Dewey Decimal Classification, 1138
- Diagnostic coaches and tutors, ICAI, 154
- Diagnostics, industrial automation, 38
- DIAGRAM grammar, 143
- Dialogue systems, 587–588
  - discourse understanding, 243
  - hermeneutics, 368
  - human-computer interfaces, 386–387
  - natural-language understanding, 672–675
    - meaning structure, 30
- DIALOG system, 420
- DIAMOND grammar, 143
- Dictionary viterbi algorithm, 86
- Difference ordering, means-ends analysis, 579–580
  - invariances, 583
- Differential equation metaphor, early vision, 1148
- Differentiation:
  - early vision, 1148
  - edge detection and, 1134
  - image understanding, 392
- Digit span, 432
- Digit symbol, intelligence and, 432
- Digital Equipment Corporation Vax hardware, 525–526
- Digital input-output, industrial robots, 947
- Digitalis Therapy Advisor, 586–587
  - functions, 588
- Dijkstra's algorithm:
  - branch-and-bound search, 1004
  - path planning, 710
  - route planning, mobile robots, 960
- Dilthey, W.:
  - hermeneutics, 363–364
  - social science, 369–370
- Dimensions, legal analysis programs, 460
- Dirac delta function, 904
- Direct techniques in clustering, 105–106
- Directed acyclic graphs (DAG), 422–423
- DISCON system, 107
- Discontinuity, edge detection and, 1134–1135
- Discontinuous switching rule, robot control, 918
- Discourse constituent unit (dcu), 237
- Discourse History Parse Tree, 238
- Discourse Representation Structures (DRSs), 329
- Discourse understanding, 233–244
  - case grammar, 334
  - computational linguistics, 144
  - dialogue phenomena, 672–675
  - hermeneutics, 368
  - information retrieval, 421
  - limits of artificial intelligence, 495
  - linguistic competence and performance, 505
  - machine translation, 564
  - natural language modes, 243–244
  - plan recognition, 241–242
    - dialogue systems, 243
    - pragmatic perspective, 241
  - speech acts, 241–242
  - speech events, 242–243
  - predicate logic, 540
  - semantic grammar, 352
  - speech act theory, 1064–1065
  - structure, 234–235
    - coherence, 235–236
    - context space theory, 236–237
    - dynamic discourse model, 237
    - modeling complexity, 234–235
    - pronoun resolution, 236
    - recent trends, 235–236
    - theory, 237
  - text meaning, 238–241
    - anaphora, 239
    - background knowledge and plausible inference, 239–240
    - logical formalisms, 239
    - story summarization, 240–241
    - true conditions for sentence and text, 238–239
  - written language, 243–244
- Discourse Units (DU), 237–238
- Discovery in machine learning, 481–485
  - description and explanation, 485
  - qualitative laws, 483–484
  - quantitative laws, 484–485
  - taxonomy formation and conceptual clustering, 481–483
- Discrete-event simulation, 757
- Discrimination networks:
  - BB1, 79
  - cognitive modeling, 113
  - hermeneutics, 373
  - natural language generation, 649
  - word-expert parsing, 707
- Discrimination tree, shape representation, 863–864
- Disease frame, development of, 587–588
- Disjointness property in clustering, 108
- Disks, associative, 17
- Disparity, stereo vision, 1084
- Display windows, human-computer interaction, 385
- Distance transform, feature extraction, 301
- Distinctiveness, episodic memory, 278
- Distributed decision-making, word-expert parsing, 705
- Distributed problem solving, 245–260
  - beliefs and concurrency, 249
  - connectionism, 202
  - contract networks, 248
  - empirical investigation, 249–250
  - functionally accurate corporate networks, 248

- global coherence with decentralized control, 247–248
- incomplete and inconsistent information, 247
- key issues, 247–248
- organizational structuring, 248–249
- scientific community metaphor, 249
- task decomposition, 247
- uses, 246–247
- Distributed processing, industrial robots, 949
- Distributed Vehicle Monitoring Testbed, 250
- District Three Computer School (NYC), 181–182
- Divisive techniques in clustering, 105
- DOCTOR program, Turing test, 1128
- Document retrieval, law applications, 457
- Domain knowledge, 251–253, 390–391
  - case-based reasoning, 478
  - circumscription, 100, 102
  - concept learning, 472–473
  - consultation programs, 252
  - first-generation expert systems, 252
  - heuristics, 377
  - image understanding system, 390, 407
  - knowledge engineering, 288
  - languages and environments, 252–253
  - natural-language automatic programming, 30–31
  - programming assistants, 786
  - phi-naught system, 787–788
  - qualitative, 251
- Domain models:
  - commonsense reasoning, 834–835
  - quantities, 834–835
  - naive physics, 747
- Domain specification, problem solving, 767–768
- Domain systems:
  - alteration, natural language interface, 659
  - early vision, 1133
- Dominance, branch-and-bound search, 1002–1003
- Don't-know nondeterminism, 556
- Dot-pattern analysis, 253–256
  - clustering, 255–256
  - grouping, 254–255
  - perceptual organization, 254
  - work in psychology, 255
- Doxastic logic, 537
  - belief systems, 64
  - heuristic theories, 67
- DRACO programming assistant, 788
- Dray, J., 1049–1059
- Dreyfus, Hubert, 740–742, 1053
  - phenomenology, 732–733
- Drilling Advisor system, 290–291
  - knowledge engineering, 297
- D-SCRIPT, belief spaces, 65
- Duchess program, 160
  - end game play, 167
- DUCK system and belief revision, 58
- Duffy, G., 362–374
- Duncker, K., 115
- DWIM program, 523
  - programming environment, 795
- Dworkin, G., 436
- Dworkin, R., 457
- Dyer, M., 980–992
- Dynamic data typing, computer systems, 175–176
- Dynamic discourse model (DDM), discourse understanding, 237
- Dynamic logic, 619
  - temporal reasoning, 874
- Dynamic modeling, path planning and obstacle avoidance, 708–709
- Dynamic programming:
  - branch-and-bound search, 1003–1004
  - matching template, 576–577
  - route search technique, 43
  - see also* Backtracking, dependency-directed
- Dynamic scope, LISP variables, 512
- Dynamic storage allocation, control structure, 213
- Dynamic worlds planning, 757
- Dynamics, robot-control systems:
  - inverse, 907
  - omissions, 911–912
  - rigid-body model, 910–911, 920–921
- DYPAR system:
  - expectation-driven parsing, 700–701
  - natural language understanding, 668
- Eager execution, logic programming, 555
- Early, J., processing, bottom-up and top-down, 780
- Earley's algorithm, 325, 327–330
- Ebcioğlu, K., 639–640
- Echoic memory, 276–277
- Economy and conceptual dependency, 197
- Edge detection, 257–266
  - artificial intelligence, limits of, 493
  - color vision, 129
  - complexity, 1138
  - connectionism, 200
  - constraint satisfaction, 209
  - differentiation, 1134
  - discontinuity and, 1134–1135
  - dot-pattern analysis, 254–255
  - early vision, 1131
    - Roberts's system, 1132–1133
  - feature extraction, 300–301
  - intensity changes, 257–266
    - directional or nondirectional operators, 264
    - multiple scales, 264–265
    - one-dimensional detection, 257–261
    - recovering physical world properties, 266
    - smoothing and differentiation, 258–261
    - spatial filters, 264
    - two-dimensional detection, 261–262
  - intensity discontinuities, 1145
  - landmark recognition, 41
  - matching template, 577
  - primal sketches, 1142
  - region-based segmentation, 880
  - robotics:
    - range image processing, 934
    - research issues, 935
  - scale and, 1135
  - SHAKY robots, 1039
  - shape analysis, 1047
- stereo vision feature acquisition, 1086
- surface constraints, 1138
- visual depth map reconstruction, 1156
- see also* Segmentation
- Edge-following algorithm, path planning, 710
- Edge orientation, Hough transform, 383
- Education applications of AI, 267–271
  - discovery environments, 269
  - expert model, 269–270
  - games, 269
  - instructional intervention, 271
  - intelligent instructional system components, 267–268
  - learning environment, 268–271
  - student model, 270
  - tutoring issue generator, 270–271
- Education, computers in, 181–185
  - alienated *vs.* syntonic learning, 184
  - computer culture, 182
  - curriculum transformation, 183
  - density, 181
  - educational technology, 183–184
  - ideology, 181
  - instruction *vs.* development, 181
  - instruction manuals, videodisks and programs, 182
  - learner control of, 183–184
  - locus of learning, 182–183
  - social interaction, 184–185
  - structured, metacognitive and developmental perspectives, 181–182
- Effective-history, 365
- Eidetic phenomenology, 373
- Eight Puzzle, 995
  - best-first search, 997
  - heuristics, 377–378
- Eight-Queens problem search methods, 994–995
- Electrocardiograms (ECGs), 727
- Electroencephalograms (EEGs), 727
- Electromagnetic radiation spectrum, 124–126
- Electron density map (EDM), CRYSLIS system, 76–77
- Electronic mail, 682
- Electronic text, 533
- ELI system, 272
  - expectation-driven parsing, 697–698
  - natural-language understanding, 663
  - script applications, 984
  - conceptual dependency, 987
- ELI-2 system, expectation-driven parsing, 698
- ELIZA system, 272
  - computational linguistics, 135–136
  - human-computer interaction, 384
  - natural-language understanding, 663–664
  - pattern matching, 717–718
  - semantics and syntax, 1024–1025
  - Turing test, 1128
- Elliot and Lesk's algorithm, 866
- Ellipsis, 272–273
  - case frame resolution, 673–674
  - natural-language interface, 658
  - natural-language understanding, 658, 673
  - semantic grammar, 352
- EL system and belief revision, 58–59
- Elusive Edge Operator, 1145

- EMACS editor, 529
- Emotion modeling, 273–274  
  commonsense reasoning, 834
- Emotionality, episodic memory, 278
- Empathetic understanding, 363–364
- Empiricism:  
  artificial intelligence and, 742  
  causal reasoning, 832  
  distributed problem-solving, 249–250  
  machine learning, 742
- EMYCIN, 95, 275, 362  
  domain knowledge, 252  
  knowledge engineering, 296  
  law applications, 460  
  logic programming, 553
- End effector, industrial robots, 945–946
- Endorsements, uncertainty, 859
- Energy landscape, early vision, 1140
- Engineering:  
  concept learning, 185  
  form atomation, 36  
  hierarchical composition, 37  
  integration, 37–38
- English language and natural language interface, 656
- Enterprise system and distributed problem-solving, 250
- Enumeration programs for chemical structure, 95
- Environment:  
  domain knowledge, 252–253  
  intelligence and, 436–437  
  learning, 184–185
- ENVISION program, 747
- Envisioning:  
  causal reasoning, 830  
  physical reasoning, 869  
  qualitative physics, 811–812  
  temporal reasoning, 872
- EP programmer's helper, automatic programming synthesis, 24
- EPAM system, 275  
  cognitive modeling, 113  
  rule-based systems, 970
- Epipolar constraint  
  obstacle detection, 40  
  stereo vision, 1085, 1087
- Episodic memory, 275–280, 593  
  expectation-driven parsing, 700  
  learning by analogy, 476–477  
  memory organization packets (MOPs), 591  
  memory proper, 277–278  
  primary memory, 276–277
- Epistemology, 280–286, 537  
  AI and, 284–285, 740  
  belief system, 64–65  
  central problems, 280–281  
  certainty, 281  
  commonsense reasoning, 833  
  cybernetics, 226–227  
  hermeneutics, 374  
  history, 281–284  
    classical period, 281  
    enlightenment, 281–282  
    Kant, 282  
    twentieth century, 282–283  
  justification, 280–281  
  knowledge sources, 280  
  philosophical questions, 736  
  predicate logic and, 538  
  representation, 281  
  story analysis, 1092  
  word-expert parsing, 703
- EPISTLE system, 287  
  office automation, 140
- Equi-NP deletion, 666
- Ergative paradigm, case grammar, 336
- Ergodicity assumption  
  texture analysis, 1104
- ERMA program:  
  natural-language generation, 646–647  
  text generation, 144
- Erman, L., 362
- Ernst, G., 318, 578–584
- Erotetic logic, 537  
  predicate logic and, 538
- Error-driven systems:  
  robot arms, 904–909
- Error estimation in pattern recognition, 726–727
- Errors:  
  computer chess programs, search and knowledge, 169  
  detection, 85  
  diagnosis rules in ICAI, 154  
  language acquisition in cognitive models, 449–450  
  language competence-performance dichotomy, 506  
  messages in natural-language generation, 642–643  
  PROLOG program synthesis, 28  
  robotics development, 924  
  sensor mapping, 41
- ERRSET program, MacLisp, 523
- ETHER language, distributed problem solving, 246
- Etherington, D., 309
- Ethics, cybernetics, 226
- Euclidean distance in color vision, 128
- EUFID system, database interface, 138
- Euler's theorem, perceptrons, 730
- EURISKO, 287  
  concept learning, 187  
  heuristics, 377–380, 476  
  industrial robots, 951
- EUROTRA system, 569
- EVAL function:  
  Horn-Clause programming, 549  
  LISP development, 520–521  
  MacLisp, 522  
  special forms, LISP 1.5, 521
- EVALQUOTE, 522
- Evaluation:  
  best-first search, 998  
  checkers programs, 88–89  
  computer chess, 168  
  coroutines, 219, 222  
  game playing:  
    importance, 317–318  
    search reduction, 316  
  heuristics, 377–379  
  information retrieval effectiveness, 421  
  LISP, 516  
  medical advice systems, 586–587  
  search function, 996–997
- Evans's Analogy Program, 732  
  law applications, 457
- EVAR system, speech understanding, 1081
- Event implies event schema, autonomous vehicle learning, 43–44
- EVENT system, frame theory, 307
- Events, episodic memory, 277–278
- Evidence:  
  uncertain, Bayesian decision methods, 51–52  
  virtual (intangible), Bayesian decision methods, 52
- Evidential reasoning, 858  
  Bayesian decision methods, 55  
  commonsense reasoning, 834  
  image understanding, 399–400  
  uncertainty, 854
- EXCAP system, computer-integrated manufacturing, 172
- Excitation and connectionism, 202
- Existential quantifier, predicate logic, 539
- Existentialism, hermeneutics, 364–365
- EXMAT, 95
- Expectation-driven parsing, 696–701  
  AD-HAC, 699  
  ATRANS structure, 697  
  CA (conceptual analyzer), 698–699  
  DYPAR/BORIS, 700  
  ELI system, 697–698  
  ELI-2, 698  
  future trends, 700–701  
  image understanding, 392–393  
  IPP (integrated partial parser), 699–700  
  micro ELI, 699  
  MOPTRANS, 700  
  MTRANS structure, 697  
  problems and solutions, 699–700  
  PTRANS structure, 697  
  scripts, 984–985  
  word-expert parsing, 699
- Expected utility measure, Bayesian decision methods, 55
- Experimental psychology, cognitive modeling, 111–114
- Expert Chromatography Assistance Team (ECAT), 95–96
- Expert-level reasoning, commonsense reasoning, 833
- EXPERT medical advice system, 95, 585–586
- Expert model, education applications of AI, 269–270
- Expert systems, 287–298  
  artificial intelligence, 14  
  case-based reasoning, 477–478  
  chemistry, 93–94  
    x-ray powder diffraction, 95  
  cognitive modeling, 114  
  cognitive psychology, 116  
  cognitive science, 122  
  computer-aided design, 153  
  credit-assignment method, 475  
  development, 10  
  distributed problem-solving, 246  
  electronics fault diagnosis, 610  
  explanation facilities, 298–300, 600  
  first-generation, 252  
  industrial automation, 38–39  
  industrial robots, 950

- inference, 418
- information retrieval, 421
- intelligent computer-aided instruction, 155
- knowledge representation, 883
- limits of, 499–500
- literature, 533
- logic programming, 552–554
- machine learning, 464
- medical advice systems, 584
- meta-knowledge and meta-reasoning, 599–600
- office automation, 680
- pattern-directed retrieval, 717–718
- pattern matching, 716
- phenomenology, 733–735
- problem reduction, 766
- programming assistants, 786
- programming environment, 790, 792
- PROSPECTOR, 797
- prostheses for visually impaired, 800
- qualitative physics and, 807
- question answering, 814–815
- primitives, 760–761
- reasoning, 822–823
- robotics planning, 941
- ROSIE system, 963
- self-reference, 1006
- social issues of AI, 1050–1052
  - joblessness, 1054–1055
- speech understanding, 1079
- Turing test, 1129
- uncertainty, 854
- understanding, 372–373
- Expertise acquisition, problem solving, 777–778
- EXPLAIN program, music applications, 638
- Explaining Bayesian decision methods, 53–54
- Explanation, 298–300
  - AGE system, 78
  - agenda-based systems, 4
  - approaches, 299
  - causal, 836
  - critiquing approach in medical advice systems, 586–587
  - current issues, 299–300
  - discovery and, 485
  - knowledge system techniques, 289–290, 540–541
  - medical advice systems, 586
  - meta-knowledge, 600
  - question answering, 815
  - rule-based systems, 968
- Exploratory programming, 789
  - INTERLISP, 794–795
  - pattern recognition, 721
- Explorer LISP machine, 529
- Expressibility in predicate logic, 540–541
- Extended standard theory (EST)
  - linguistic competence, 503–504
  - transformational grammar, 355, 357, 360
- Extensibility of programs, programming environment, 795–796
- Extraction and pattern recognition, 720
- Extragrammatical utterances, 671–672
- Extraposition grammar (XG) technique, 339, 342
  - learning, 268–271
  - programming, 789–796
- Facet theory of intelligence, 433–435
- Factoring, binary resolution inference rules, 897
- Fahlman, S., 204
- FAIM-1 architecture, 216
- Falk, G., cycle of perception, 394–395
- False-alarm rate, 722
- Falsity preservation, 188
- Fast Fourier transform (FFT) algorithm, 576–577
- FDS program, means-ends analysis, 578, 581
- Feature co-occurrence restrictions (FCRs), GPSG, 343
- Feature correspondence, motion analysis, 621–622
- Feature detection:
  - hierarchy, 1136–1137
  - stereo vision, 1086
- Feature extraction, 300
  - components, 301
  - connection machines, 200
  - geometric features:
    - diameter, 301
    - moments, 301
  - local operators, 300–301
  - skeleton, 301
  - template matching, 577
- Feature instantiation, generalized phrase structure grammar, 343
- Feature matching, stereo vision, 1087
- Feature space, color vision, 128
- Feature specification defaults (FSDs), generalized:
  - phrase structure grammar, 343
- Features in word formation, 620
- Feedback:
  - connectionism, 202
  - cybernetics and, 226
  - industrial robots, 946
  - image understanding systems, 396
  - robot arms, 904–909
- Feigenbaum, Edward A., 233, 478, 1052
- Feldman, J., 203–204
- Felicity conditions, hermeneutics, 366
- Ferguson, G., 437
- Fidelity X, computer chess, 169
- FIDO system, obstacle detection, 40
- Fifteen Puzzle, 377, 995
- Fifth Generation Computers, xxi, 175, 545
  - DADO architecture, 227–228
  - computer system example, 177, 179–180
- Fikes, R., 309
- File structure alteration, natural language interface, 659
- Fillmore, C.:
  - case markers, 336
  - deep case system, 336
- Filtering:
  - scale-space, 974
  - Waltz, 208, 1162–1165
- Fingerprinting:
  - intensity changes, 261
  - pattern recognition, 728
- Finite identification, 415
- Finite-state automation (FSA):
  - ATN parsing, 326, 693
  - connectionism, 201–202
  - deterministic (DFA), 693
- inductive inference, 414
- nondeterministic (NFA), 693
- word formation, 620
- Finite-state grammar, 689
- Finite-state transition diagram, 326
- First-generation expert systems, domain knowledge, 252
- First lady concept, 192
- First-order logic:
  - control structure, 214
  - inductive inference search, 413
  - inheritance hierarchy, 425
  - nonmonotonic reasoning, 852
  - procedural semantics, 1027–1028
  - program (FOL), 302
  - propositional logic and, 558
  - reasoning, 823
  - self-reference, 1006
  - theorem proving, 1115
- First-order predicate calculus (FOPC):
  - binary resolution, 895
  - control structure, 215
  - planning, 749
  - semantic networks, 1013
  - extensions, 1020–1021
- First-order predicate logic:
  - default reasoning, 840–841, 844
- FIS (Fault Isolation System), 610
- Fischler, Martin A., 1083–1089
- Fixnums:
  - data structure, 518
  - LISP, 510–511
  - run time typing, 517–518
- Flavors software:
  - control structures, 212–213
  - LISP machine, 526, 529
- Flexible manufacturing system (FMS):
  - industrial robots, 949–950, 954
- Flight simulation, texture analysis, 1102
- Flores, F., hermeneutics, 372–373
- Flowcharts:
  - program synthesis, 22–23, 29
  - construction revision, 24–25
- Flow fields, motion analysis, 626–627
- FLOW programming tutor, 155
- FLPL (Fortran list processing language), 520
- Focus of discourse understanding, 144
- Fodor, J. A., 284
- FOL (first-order logic) program, 302
  - self-reference, 1007–1008
- Foley, J., 387
- Folk psychology, frame theory, 303–305
- Following, path planning and obstacle avoidance, 713–714
- Fool's Disks problem, 581–583
- Foot feature principle, GPSG, 343
- Foreign languages and machine translation, 133
- Foreshortening, texture analysis, 1110–1111
- Forgetting and creativity, 224
- Forks, game playing, 315–316
- Formal decision analysis theory, 229
- Formal differentiation, limits of, 502
- Formal logic:
  - deep case grammar, 335
  - image understanding, 398–399
  - language theory, 326, 354
  - limits of AI, 494
  - nonmonotonic logic, 850

- predicate logic, 540
- reasoning and, 848
- Formal Principles of Language Acquisition*, 445
- FORMAT statement, character recognition, 85
- Formation characteristics and problem solving, 775–777
- FORTH language, industrial robots, 950
- FORTTRAN:
  - control structures, 213
  - knowledge engineering, 297
  - knowledge source organization, 84–85
  - LISP development, 518–519
  - programming environment, 792
  - semantic networks implementation, 1022
  - SLIP utility, 1048–1049
- Forward chaining:
  - expectation-driven parsing, 696
  - image understanding, 398
  - knowledge representation, 887
  - military applications, 608
  - path planning and obstacle avoidance, 709–710
  - pattern-directed retrieval, 718
  - pattern matching, 719
  - planning operation, 753
  - processing, bottom-up and top-down, 780
  - reasoning, 824
  - rule-based systems, 968
- Forward-looking infrared (FLIR) imagery, texture analysis, 1106
- FOUND operator, beam search, 57
- Foundationalism, 281
- Fourier domain, matching template, 576
- Fourier transforms, template matching, 577
- FR system, belief revision, 60
- Fractal sets, 1110
  - scale-space methods, 974–975
- FRAIL (Frame-based AI Language), 309
- Frame axiom:
  - logic programming, 552
  - planning, 750
- Frame-based description language, domain knowledge, 253
- Frame-based representation, semantics, 1028
- Frame difference, 2-D translation, 627
- Frame problem, 302
  - artificial intelligence and, 11
  - belief revisions, 58
  - causal reasoning, 831
  - knowledge representation, 883
  - logic programming, 552
  - phenomenology, 733
  - planning, 750
  - situation calculus, 871
  - temporal reasoning, 836
- Frame recognition, 304–305
  - indexing, 305–306
- Frame representation language (FRL), 308
- Frame systems, 302
  - natural language generation, 649
- Frame theory, 302–310
  - beginnings, 302
  - case frame instantiation, 668–669
  - categories:
    - basic, 306
    - subordinate, 306
    - superordinate, 306
- commonsense reasoning, 834
- default predictions, 303
- default reasoning, 842–843
- determination, 305
- discourse understanding, 233
- domain knowledge, 251
- folk psychology, 303–305
- frame-driven recognition, 304–305
- higher level knowledge, 304, 308
- hybrid systems, 309
- image understanding, 398–399
- information retrieval, 421
- inheritance, 423
  - programming language, 430
  - slots, 426
- intent, 303
- invocation, 305
- ISA hierarchies:
  - cases, slots and predictability, 308
  - categories, 306–307
  - epistemological viewpoint, 307
  - language subsumption and expressiveness, 308
  - slots and property inheritance, 307–308
  - virtual copy, 307–308
- knowledge representation, 885
- knowledge system construction, 294
- languages, 303
- legal analysis programs, 460
- limits of artificial intelligence, 495–496
- manipulation, 305–306
- medical advice systems, 585, 587
- memory organization, 302, 305–306
  - frame size, 306
- methods, 309–310
- military applications, 606–608
- misrecognition, interpretation and reinterpretation, 304
- organizational features used in languages, 308–310
  - object-oriented language, 309
  - uniformity, coherence and expressiveness, 309
- phenomenology, 732, 741–742
- primitives, 760
- predicate calculus, 308–310
- predictions, 304
- property inheritance, formalizing, 309
- reasoning, 826
- recognition, interpretation and prediction, 304
- room frame analysis, 303
- script applications, 984
- semantic networks, 1011
- semantics, 308
- state variables, 309
- terminal sharing, 303
- terminology, 302–303
- uncertainty and, 854–855
- Frames:
  - cognitive psychology, 116
  - stereotyped social situation, 838
- Franklin, J., 604–613
- Franz LISP, 448, 525
  - history, 526
  - word-expert parsing, 703
- Frederiksen, J., 436
- Fredkin, Edward, 160
- Fredkin Prize, 160
- Free boundary-values in early vision, 1148
- Free logic, 537
  - predicate logic, 541
- Free-space algorithms, path planning and obstacle avoidance, 711–712
- Frege, Gottlob, 282–283, 441
- Frequency-domain analysis in robot-control systems, 903
  - inverse dynamics, 907
- Frequency identification, inductive inference, 415
- Freud, Sigmund and emotion modeling, 274
- Freuder's algorithm for constraint satisfaction, 209
- Frey, P. W., 380–381
- FRL (Frame-oriented representation language), 312
  - AKO relation, 422
  - inheritance hierarchies, 422, 425
  - knowledge representation, 882–883, 888
- FRL-O program, 312
- FROB program, 869
- FRUMP system:
  - discourse understanding, 240
  - natural-language understanding, 663
  - scripts, 981
  - natural language understanding, 988
  - story analysis, world knowledge acquisition, 1096
- Fully automatic high-quality translation (FAHQT), 133
- FUNARGS:
  - control structures, 212
  - LISP 1.5, 521
- Function definition, LISP variables, 511
- Functional electrical stimulation (FES) for prostheses, 797–798
- Functional grammar, machine translation, 570
- Functionalism:
  - AI and, 737–738
  - intentional stance, 738
  - logic programming, 550–551
  - parallelism, 555–556
  - physicalism and, 737–738
- Functional unification grammar (FUG):
  - natural language generation, 650–651
  - recursive layers, 653
- Functionally accurate corporate network for distributed problem solving, 248–249
- Fuzzy logic:
  - default reasoning, 845–846
  - machine translation, 567–568
  - reasoning, 823
  - uncertainty and, 854, 862
  - necessity and probability theory, 858
- Fuzzy set theory:
  - commonsense reasoning, 834
  - concept learning, 192
  - concurrent PROLOG, 556
  - default reasoning, 845–846
  - knowledge representation, 889
  - literature, 533
  - pattern recognition, 721
  - social issues of AI, 1049–1059
  - speech understanding, 1077
- Gabriel, R., 508–527
- Gabriel graph, dot-pattern analysis, 255

- Gadamer, H.:  
 Habermas debate with, 366–368, 370  
 philosophical hermeneutics, 365–366, 372
- Gain scheduling, robot control, 919
- Galton, Francis (Sir), 431–432
- Game playing, 88, 312–318  
 AND/OR graph, 7–8  
 education applications of AI, 269  
 five-deep force, 318  
 game trees and, 313  
 history, 9–10  
 horizon effect, 380–381  
 importance of evaluation and learning, 316–318  
 learning as, 15  
 literature, 530  
 mathematical formulation, 312–314  
   Conway approach, 314  
 minimax procedure, 614–617  
 partition representation, 313  
 perfect-information, games, 319–321  
 reasoning, 822, 848  
 search reduction, 315–316, 994  
   evaluation, 316  
   forks, 315–316  
   kernels, 315–316  
   Koffman-Citrenbaum technique, 315–316  
   stability, 316  
 strategies, 313–315  
 two-person, 312  
   game trees, 319  
   heuristics, 378–379
- Game trees, 319–321  
 AND/OR graphs, 319–321  
 branch-and-bound search, 1003  
 checkers, 88  
 computer chess, 162  
 game playing, 313  
   equivalent nodes, 313–314  
   minimax value of nodes, 313  
 horizon effect, 380–381  
 leaf nodes, 319  
 minimax procedure, 614–617  
   backed-up values, 5  
 reasoning, 822  
 recursion, 875–876  
 roots, 319  
 search, 995
- Gap programs, inductive inference, 416
- Garbage collection:  
 control structure, 213  
 LISP machine, 525–526, 529  
   run time type, 518  
   word-expert parsing, 705–706
- Symbolics 3600, 177  
 ephemeral, 216
- Gardner, A. V. D. L., 456–461
- GARI program, 754, 756
- Garrigan, S., 944–954
- Gaussian distribution, pattern recognition, 722–723
- Gaussian image:  
 extended, 835  
   shape analysis, 1041, 1046–1047  
 intensity change detection  
   one-dimensional, 258, 261  
   two-dimensional, 263  
 scale-space methods, 974–976  
 texture analysis, 1104
- Gazdar, G., 342–343, 347–348
- Gelade, G., 116
- Gelernter, Herbert, 520
- Geller, J., 272, 362, 563, 603–604, 1048–1049, 1099
- General AI machine, NON-VON computers, 679
- General Problem Solver (GPS), 9, 114, 323, 537  
 AND/OR trees, 580  
 law applications, 457  
 means-ends analysis, 578–581  
 pattern matching, 719  
 planning, 757  
 propositional logic, 562  
 reasoning, 822  
 reduction schema, 775  
 REF-ARF programs, 877  
 rule-based systems, 970  
 self-reference, 1006
- General-to-specific method in heuristics learning, 475
- Generalization:  
 exemplary, inference, 419  
 inductive inference search, 413–414  
 memory organization packets, (MOPs), 591  
 motion analysis, 3-D, 626–627  
 virtual machine control structure, 213–214
- Generalized cone (GC), 321
- Generalized cylinders, 321–323  
 computer vision, 323  
 issues, 321–323  
   adequacy, 322–323  
   complete specification, 321–322  
   continuity, 322  
   detail levels, 323  
   disjoint, 322  
   formalization, 322  
   interior, 322  
   product decomposition, 322  
   similarity and object class, 322  
   structure, 322  
 high level vision, 1149  
 multisensor integration, 633  
 path planning and obstacle avoidance, 712  
 robotics range image processing, 934–935  
 shape analysis, 1046  
 shape description, 835  
   volumetric descriptions, 863–864
- Generalized phrase structure grammar (GPSG), 342–343  
 natural-language understanding, 665  
 parsing, 1021  
 revival, 347–348  
 transformational grammar, 354–355, 359
- Generalized potential fields (GPFs), obstacle avoidance, 42
- Generalized stereo paradigm, 1083
- Generalized transition networks, 331
- Generate-and-test method  
 causal reasoning, 830  
 problem reduction, 763  
 problem solving, 769–770, 776–777  
   initialize, 769
- Generation:  
 language, conceptual dependency, 198–199
- natural-language, 642–654  
 character, 643–644  
 common approaches, 647–650  
 components and terminology, 644–645  
 grammar treatments, 650–653  
   functional unification grammar (FUG), 650–651  
   surface structure as intermediate representation level, 651–652  
   systemic grammar and ATNs, 652–653  
 historical perspectives, 646–647  
 lexical choice, 649  
 message refinement, 647–649  
 natural-language speech acts, 1063  
 phrasal lexicons, 649–650  
 planning, 653–654  
   psycholinguistic theory, 653–654  
 semantic networks, 1021  
 state of the art, 645–646
- Generative paradigm, 354
- Generic operations, Symbolics 3600 computer system, 176–177
- Genetic-algorithm approach, machine learning, 467
- GENSYM system, LISP extension, 526  
 semantics, deep structure, 231  
 solutions, problem solving, 769
- Geometry:  
 analogical reasoning, 476–477  
 depth map analysis, 1158–1159  
 feature extraction, 301  
   compactness, 301  
   elongatedness, 301  
 LOGO system, 269  
 modeling, limits of, 497  
 range image processing, 934  
 shape analysis, 835, 1041
- GEORGE chemical tutor, 96
- Gerberich, Carl, 520
- Gestalt psychology:  
 dot-pattern analysis, 254  
 early vision, 1147–1148
- Gestures, pattern recognition, 384
- GETA system, 569
- Gettier, E., 283
- G grammar, 341–342
- Gibson, J. J., 116
- GIMADS (Generic Integrated Maintenance Diagnostics), 611
- Gini, M., 716–719, 762–766
- Gist specification paraphraser, 785
- GLAUBER system, 107, 484
- Glitter programming assistant, 787
- Global-positioning satellites (GPSs), autonomous vehicles, 41–42
- Goals:  
 classification, 837  
 dependency network, 107  
 identification, natural-language generation, 643–644  
 planning, 748  
   bonus, 753  
   embedded, 754  
   hierarchies and ordering, 755  
   protection, 750  
 processing, bottom-up and top-down, 780  
 satisfaction, image understanding, 393  
 states, machine learning, 475  
 story analysis, 1094



- see also* specific types of goals, e.g. Transform goals  
 Goal trees, story analysis, 1094  
 Godel's theorem, 739-740  
   epistemology, 283-284  
   propositional attitudes, 837  
   self-reference, 1005  
 Goffman, E., 302  
 Goldbach's conjecture, 224  
 Golden, Jeff, 523  
 Goldstein, Ira, 312  
 Goldstein, M., 318, 445  
 Go-Moku game:  
   kernels on, 315-316  
   learning and, 317  
   search reduction, 315  
 GO system:  
   game trees, 319  
   strategy efficiency, 312  
 Government-binding theory, 507  
   transformational grammar, 355, 359  
 GPS, *see* General Problem Solver  
 GPSS simulation language:  
   computational linguistics, 140  
   natural language program synthesis, 32  
 Gradient dynamics, robot-control systems, 914-916  
 Gradient operator, early vision, 1133  
 Gradient space, shape analysis, 1041  
 Grammar:  
   acquisition, machine learning, 479  
   augmented transition network (ATN), 323-332  
     cascaded, 331-332  
     factoring, 325  
     formal properties, 329  
     generalized transition networks, 331  
     history, 325-326  
     human-computer interaction, 384  
     linguistic experimentation, 328-329  
     misconceptions, 330-331  
     parsing, 327, 330  
     specification, 327-328  
     *see also* ATN grammar  
   and Grammar, augmented transition network base, 355-356  
   case, 333-338  
     cognitive modeling, 113  
     deep cases  
       grammatical explanation, 333-334  
       meaning representation, 334-335  
     event description, 335  
     propositions, 335  
     sentence meaning, 335  
     surface cases, 333  
     systems, 335-338  
       Celce, 336  
       Fillmore, 336  
       Grime's, 336-337  
       Schunk, 337-338  
   cognitive models of language acquisition, 448-449  
   complex feature-based, 692  
   computational linguistics, 141-143  
   context-free, 326-327  
   context-free phrase structure (CF-PSG), 343  
   definite clause, 142, 339-342  
     ATN perspective, 329-330  
     Colmerauer analysis, 340-342  
     extensions of, 342  
   G grammar, 341-342  
   logic grammars, 339-340  
   natural language analysis, 340  
   EPISTLE system, 287  
   extraposition (XG), 342  
   formalism developments, 136-137  
   functional unification (FUG), 650-651  
   generalized phrase structure, 342-343  
     revival, 347-348  
   generative, 354  
   inductive inference method, 412  
   language acquisition:  
     negative instances, 481  
     sentence-meaning pairs, 480-481  
     simple sentences, 479-480  
   lexical-functional, 359  
   linguistic case, frame theory, 308  
   metamorphosis (MG), 339  
   modifier structure (MSG), 342  
   natural language generation, 644-645, 650-653  
   parsing and, 687-688  
   phrase-structure, 344-350  
     formal properties, 345-346  
     generalized, 347-348  
     revival, 347-348  
     terminal symbols, 346  
     transformation grammar, 346-347  
     tree-adjointing grammar (TAG), 348-350  
   trees, 344  
     revival, 347-348  
   problem solving, 768-769  
   puzzle (PG), 342  
   semantic, 331-332, 350-353  
     advantages, 351-353  
     discourse phenomena, 352  
     efficiency, 351-352  
     habitability, 352  
   case frames, 338  
   limitations, 353  
   passive relationship, 351  
   semantic networks, 1021  
   structure, case frames, 338  
   systemic, 372  
     natural-language generation, 652-653  
   transformational, 134, 353-361  
     computational models, 355-356  
       Marcus parser, 360-361  
       Petrick system, 360  
       sentence analysis, 360  
   derivation process, 355-356  
   generative grammar, 354  
   language, 354  
   syntactic and semantic rules, 354  
   variations, 355-359  
     alternative theories, 359  
     base grammar, 355-356  
     components, 356-357  
     constraints, 357-358  
     extended standard theory, 357  
     government-binding theory, 359  
     X-bar theory, 358-359  
   tree (TG), 342  
   unification-based, 692  
   *see also* Types of grammar  
   Grammatical inference, 411  
   Grammatical patterning, 505-506  
   Granularity, uncertainty, 860  
 Graphics:  
   computerized in chemistry, 94  
   interface, programming tools, 793  
   robot-programming systems, 947  
   shape representation:  
     boundary descriptions, 864  
 Graphs:  
   A\* algorithm, 1  
   AND/OR, 7-8  
   backward search, 12  
   Bayesian networks, 54  
   branch-and-bound search, 1002  
   chemistry programs, 93  
   directed:  
     acyclic, 422-423  
     beam search, 56  
   game playing, 313-314  
   language:  
     human-computer interaction, 385  
     natural-language interfaces, 655  
   search:  
     limits of, 497  
     mobile robots route planning, 960  
     representation, 994-995  
   semantic networks logic, 1019  
   structures, blackboard architecture, 78  
 GraphSearch, 710  
 Graph-theoretic algorithms, dot-pattern analysis, 256  
 Graph Traverser program, 999  
 GRASPER language, language acquisition, 448  
 Gray-level images:  
   robotics sensing, 933  
   texture analysis, 1105  
 Green's question-answering program, binary resolution, 900  
 Greenblat, R., 522, 528-529  
   LISP machines, 528-529  
 Greenblatt Chess Program, 564  
 Greibach:  
   normal-form grammar, 689-690  
 Griffiths, T. V., 691  
 Grimes deep case grammar system, 336-337  
 Griss, Martin, 524  
 Gross shape descriptors, multisensor recognition, 634  
 Grossman, D., 35-39  
 Grosz, B. J., 237  
 Grouping:  
   dot-pattern analysis, 254-255  
   early vision, 1147-1148  
   high level vision, 1149  
   information retrieval, 419-420  
 Group technology:  
   computer-aided design, 153  
   industrial robotics, 950  
 Group theory, theorem proving, 1115  
 Guard's program, binary resolution, 900  
 GUIDON system, 155, 362  
   human-computer interaction, 385  
 Guilford, J., 433-434  
 Gunderson, K., Turing test, 1127  
 GUS system:  
   agendas in, 4  
   office automation, 140  
   question answering, 818  
   status labeling procedure, 319-320  
   terminal node, 319  
   winning nodes, 315, 319-321

- Habermas, J.:  
 Gadamer debate with, 366–368, 370  
 hermeneutics, 366, 369
- Habitability, natural-language interface, 657–658
- HACKER system, 362, 602
- Hall, H., 731–735
- Halliday, M. A. K., systemic grammar, 653
- Halpern, J. Y., heuristic belief system theory, 68
- Halting problem:  
 inductive inference, 410  
 logic programming, 555
- HAM-ANS system:  
 cognitive modeling, 113  
 computational linguistics, 139  
 question answering, 818
- Hamiltonian systems:  
 robot-control, 914–915  
 gradient systems integration, 915–916  
 stability, 915–916
- Handbook of Artificial Intelligence*, 302–303
- Harder, D. S., 35
- Hardt, S. L., 194–199, 747
- Harmon, S. Y., 39–44, 957–961
- HARPY system, 362  
 beam search, 57  
 human-computer interaction, 384  
 image understanding, 398  
 beam model, 397–398  
 speech recognition, 1068–1069  
 speech understanding, 1079–1081
- Harris, L. R., 443
- Harris, Z., 566
- Hart, H. L. A., 457
- Harvard Syntactic Analyzer, 142, 566
- Hash tables, computer chess, 164–165  
 memory tables, 167–168
- HASP (Heuristic Adaptive Surveillance Project), 475, 607–608  
 blackboard architecture, 73, 76  
 rule-based systems, 971
- HAWKEYE system, 399, 401  
 knowledge engineering, 297
- Hayes, P., 760
- Hayes, Patrick:  
 belief systems, 63  
 epistemology, 284  
 naive physics, 871–872  
 neat/scruffy debate, 537
- Hayes, Philip J., 660–675
- Hayes-Roth, B., 73–79
- Hayes-Roth, F., 287–298, 973
- Hazardous events, pattern recognition, 722
- Header objects, LISP, run time typing, 517–518
- Head feature convention (HFC), GPSG, 343
- Head grammar (HG), 350
- Headrick, T. E., 457–458
- Hearn, Anthony, 524
- HEARSAY system, 118  
 domain knowledge, 252  
 human-computer interaction, 384  
 image understanding, 393, 398
- HEARSAY-II, 362  
 blackboard architecture, 73–76  
 control structures, 218  
 distributed problem solving, 249  
 knowledge engineering, 288, 296–297  
 knowledge representation, 888  
 reasoning, 848  
 rule-based systems, 971  
 speech recognition, 1069  
 speech understanding, 1079–1081
- HEARSAY-III  
 blackboard architecture, 73, 78  
 domain knowledge, 252  
 knowledge engineering, 297
- Heart waves, 727
- H-easy* function, 412–413
- Hegel, G., hermeneutics and, 365–366
- Heidegger, M.:  
 context and holism, 741  
 critiques Husserl, 371  
 hermeneutics, 364–365  
 phenomenology, 367–369, 372, 731–733
- Henderson, D. Austin, 680–683
- Hennigan School, 181
- Henschen, L., 418–419, 822–826, 1115–1122
- Hepatitis Knowledge Base project, 421
- HerbAl system, 228
- Herbrand instantiation:  
 binary resolution, 893  
 theorem proving, 1120
- Herbrand interpretation:  
 binary resolution, 896  
 theorem proving, 1116
- Herbrand models, 547
- Herbrand theorem, 1116–1117
- Herbrand universe, 547  
 binary resolution, 896  
 theorem proving, 1116–1117
- Hermeneutic arc, 368–369
- Hermeneutic circle, 363  
 ontological, 365
- Hermeneutics, 362–374  
 artificial intelligence, 370–373  
 natural language understanding, 371  
 text structure analysis, 371  
 understanding, 372–373  
 background, 362–363  
 classical methodological, 363–364  
 critical, 366–367  
 communicative action theory, 367  
 Habermas-Gadamer debate, 366–368, 370  
 strategic orientation, 366  
 universal pragmatics, 366  
 depth semantics, 368  
 eidetic phenomenology, 373  
 emancipatory science, 370  
 ideology critique, 368  
 meta science, 369–370  
 methodological social sciences and, 369–370  
 natural science, 370  
 ontological assumption, 372  
 origin of word, 363  
 paradigms for, 370  
 phenomenological, 367–369  
 hermeneutic arc–Ricoeur's theory, 368–369  
 philosophical, 364–365  
 Gadamer, 365–366  
 Ricoeur's theory of, 368–369  
 Schleiermacher and Dilthey methodology, 363–364  
 social science and, 369–370
- Heterarchy:  
 control structures, 213  
 image understanding, 396–397
- Heuristic DENDRAL, problem solving, 776
- Heuristics, 376–380  
 A\* algorithm, 1  
 artificial intelligence and, 11  
 blackboard architecture, 73  
 checkers-playing programs, 90, 92  
 color vision, 130  
 creativity, 224  
 decision making, 118  
 domain knowledge, 251  
 education applications of AI, 270  
 EURISKO program, 287  
 evaluation functions, 377–379  
 learning, 379  
 one- and two-player, 379  
 simplified models, 378–379  
 single-agent problems, 377–378  
 two-person games, 378–379  
 game playing, search reduction, 316  
 history, 163  
 horizon effect, 380–381  
 image understanding, 398–399  
 killer, 163  
 knowledge engineering, 288, 294  
 learning, open problems, 476  
 limits of artificial intelligence, 494  
 logic programming, expert systems, 553–554  
 machine learning:  
 instances to, 475–476  
 problem solving, 473–474  
 natural-language automatic programming, 32  
 problem solving, 769  
 solution methods, 770–771  
 reasoning and, 822  
 REF-ARF programs, 876–877  
 rules, 379–380, 966  
 search, 996–997  
 bidirectional, 1000  
 branch-and-bound, 1000  
 methods, 994  
 SNIFFER system, 1049  
 story analysis, 1092  
 scripts, 1093  
 Turing machines, 1125  
 uncertainty, 854–855  
 Viterbi algorithm, 1161
- Hewitt, C., 603
- Hewitt's actor model:  
 causal reasoning, 828  
 demons, 232  
 lambda calculus, 442  
 logic programming, 556–557  
 object-oriented programming, 452  
 script, 1981
- Hewlett-Packard generalized phrase structure grammar, 343
- Hex search reduction, 315
- Heyting, Arend, 283
- H-honest* function, 413
- Hierarchies:  
 autonomous vehicles, 44  
 classifiers, pattern recognition, 724  
 control strategy, mobile robots, 957

- CRYALIS system, 76–77  
 decomposition, 396  
 discourse understanding, 236  
 feature detection, 1136–1137  
 frame theory, object-oriented programming language, 309–310  
 goal, story analysis, 1094  
 of groupings, 254–255  
 image understanding, 397  
 inheritance, 422–430  
   demons, 429  
   first-order logic, 423–424  
   inheritable relations, 428–429  
   multiple inheritance, 422–423  
     exceptions, 424–426  
   parallel-marker propagation, 429  
   programming languages, 429–430  
   properties and slots, 422  
   simple exceptions, 424  
   slot constraints, 426  
   splits and partitioning, 427–428  
   structured concepts, 426–428  
   tangled, 422–423  
   taxonomy, 422  
 intelligence structure, 433–435  
 ISA  
   connectionism, 203  
   frame theory, 306–309  
   goal-directed activation, 395–396  
   inheritance, redundant, 422, 425  
 musical structure, 639–641  
 NOAH system, 677  
 planning, 754–755  
 polythetic, 483  
 route finding, 866  
 semantic networks, 1016–1017  
 tangled, 422–423  
   frame theory, 307  
   semantic networks, 1017  
 taxonomic, 422, 482  
   inheritance, 422  
 template-matching techniques, 577  
 High-level vision, 389  
 HILARE, 712  
   autonomous vehicles, 39–40  
   obstacle detection, 40  
   dead reckoning position location, 42  
   hierarchical structure, 44  
   learning, 43  
   map transforms and path graph nodes, 43  
   obstacle avoidance, 42  
   sensor mapping, 41  
 Hilbert, David, 283  
 Hilbert transform, intensity change detection, 261  
 Hildreth, E., 257–266, 687  
 Hill, J. C., 444–450  
 Hill-climbing:  
   best-first search, 999  
   checkers-playing programs, 91  
   inductive inference, 416  
   problem solving, 776–777  
   search, 997  
   SNPR system, 479  
 Hiller, L., 639  
 Hindle, D., 230–231  
 Hintikka belief systems, 64  
 Hinton, G. E., 80–81  
 Hirst, G., 1024–1029  
 Histogram analysis, Hough transform, 382  
 Histories:  
   causal reasoning, 831  
   heuristics, 163  
   temporal reasoning, 836, 871  
 Hoare's method of theorem proving, 1121–1122  
 Hogger, C. J., 544–557  
 Hohne, B., 93–97  
 Ho-kashyap procedure, pattern recognition, 723  
 HOLD actions:  
   ATN grammar, 328, 330  
   planning, 754  
 Holism, phenomenology, 741  
 Hollenberg, J., 229–230  
 Holmes, D. W., 457  
 Holyoak, K., 113–117  
 Homunculi, 738  
   spatial reasoning, 867–868  
 Hooke's law, qualitative physics, 812  
 HOPE LANGUAGE, 555  
 Hopfield networks, 1156  
 Horizon effect, 380–381  
   computer chess, 163–164  
   game playing search reduction, 316  
   negative-horizon effect, 380–381  
   positive-horizon effect, 380–381  
   terminal positions, 380–381  
 Horn, J. L., 434  
 Horn clause, 498, 546–547  
   associative memory, 18  
   computation adequacy, 551  
   control structure:  
     logic programming, 215  
   definite-clause grammar, 339  
   inductive inference search, 413–414  
   logic programming, 544  
     extensions, 548–549  
   problem reduction, 763–764  
   PROLOG, constraint satisfaction, 210  
 Hough transform, 382–383  
   accumulator, 383  
   edge detection, 1137  
   landmark recognition, 41  
   limits, 493  
   motion analysis, 623  
   parametric, 383  
   road detection, 40–41  
   road following, 959  
   shape analysis, 1044  
   template matching, 577  
 Huang, T. S., 620–630  
 Hueckel operator, intensity change, 258–259  
 Huffman-Clowes world, Waltz filtering, 1163–1165  
 Hull, J. J., 82–87  
 Human analysis, game playing and, 317  
 Human-computer interaction, 383–385  
   adaptation, 385  
   cognitive science, 120–121  
   design, 386–387  
     styles, 386  
     techniques and guidelines, 386–387  
   intelligent user interfaces and computer-aided instruction, 384–385  
   natural language applications, 383–384  
   natural language interface, 655  
   other developments, 385–386  
   pattern recognition, 384  
   prostheses, 799  
   speech, 384  
   teleoperators, 1100–1101  
 Human learning as model for machine learning, 464  
 Human stereo vision, 1084  
 Human vocal tract apparatus:  
   speech recognition, 1067  
   speech synthesis, 1070  
 Hume, David, 282  
   concept learning, 191  
 Humphreys, L. G., 433–434  
 Hunt, E., 436  
 Hussain's theorem of uncertainty, 856  
 Husserl, Edmund, 364–365, 371  
   context and holism, 741  
   epistemology, 282–283  
   phenomenology, 367–369, 731–733  
 HWIM system:  
   augmented transition network grammar, 327  
   human-computer interaction, 384  
   image understanding system, 390  
   speech recognition, 1069  
   speech understanding:  
     knowledge-source interpretation, 1079  
     processing strategies, 1081  
 Hybrid cellular-kinematic systems, 1011  
 Hybrid derivation:  
   problem solving, 775–777  
 Hydrophones, ANALYST system, 607  
 Hypergraphs, problem reduction, 763  
 HYPO program, 460  
 Hypothesis:  
   activation, image understanding, 392  
   concept learning, 187–188  
   factoring, ATNs, 325  
   formation, hermeneutics, 368  
   natural-language interfaces, 655  
   space:  
     machine learning, 466–467  
     medical advice systems, 585  
   testing, image understanding, 392  
 IBM 801, computer architecture, 215  
 IBM, LISP development, 525  
 IBM 7090, LISP development, 522  
 IC-PROLOG system, 546–547  
   coroutine, 222  
   execution, 555  
   parallelism, 556  
   subgoal selection, 555  
 ICAI, *see* Intelligent computer-aided instruction  
 Iconic memory, 276–277  
 Ideal speech situation, hermeneutics, 366  
 Identification in the limit concept, 409–416  
 Identifiers, LISP system, 511  
 Identity assumption, intensity changes, 265  
 Idioms in word-expert parsing, 702  
 ID3 system, 191  
   decision tree, 470–471  
   machine learning, 483

- If-then-else special form, LISP, 509
- If-then rules:
  - expectation-driven parsing, 696
  - military applications, 606
- Ignorance and belief systems, 66
- IGS system, image understanding, 402–403, 406
- ILIAD system, computer-aided instruction, 140
- Ill-conditioning, early vision, 1145
- Iliac Suit*, 639
- Ilurwitz, R., 362–374
- Image acquisition, stereo vision, 1083
- Image analysis:
  - artificial intelligence, 493
  - segmentation, 877–880
- Image intensity, early vision, 1142
- Image irradiance equation, shape analysis, 1041–1043
- Image matching, stereo vision, 1086–1087
- Image processing, tactile, 936
- Image representation, communication and, 1136
- Imagery:
  - analogue representation, 881
  - spatial reasoning, 867–868
  - wire-frame representation, 890–892
- Image understanding, 389–407
  - artificial intelligence, 493
  - clustering, 105
  - color vision, 128
  - control structures, 393–396
    - beam models, 397–398
    - blackboard models, 397–398
    - data-directed activation, 396
    - failure-directed activation, 396
    - goal-directed activation, 395–396
    - heterarchical models, 396–397
    - hierarchical models, 397
    - model-directed activation, 396
    - perception cycle, 394–395
    - rule-based approaches, 398
    - temporally-directed activation, 396
  - design, 390
  - distance determination, 1088–1089
  - examples for specific problem domains, 400–406
    - aerial photographs, 400–402
    - indoor scenes, 402–403
    - medical images, 404–405
    - outdoor scenes, 402
  - group information, 389
  - historical perspective and techniques, 393–399
  - inference and control requirements, 392
  - label entities, 390
  - performance evaluation, 678
  - reasoning and uncertainty, 399–400
    - planning, 400
    - relaxation-labeling process, 399
    - spatiotemporal reasoning, 400
  - representational and control requirements, 391–392
  - representational formalism, 398–399
    - frame theory, 399
    - heuristics, 399
    - semantic networks, 399
    - spatial representation, 398–399
  - research issues, 407–408
  - speech understanding, 390
  - texture analysis, 1101–1111
  - three-dimensional surfaces, 389
  - transformation of image-centered representations, 389–390
  - two-dimensional grouping, 389
  - Viterbi algorithm, 1160
- Imitation game, 1126–1127
- Immediate dominance (ID) rules, GPSG, 343
- Impedance-control, obstacle avoidance, 713
- Imperfect knowledge, minimax procedure, 614
- Impetus theory, causal reasoning, 832
- Implementation, LISP, 527
  - programming assistants, 785
  - programming environment, 791
- INBR system, computational linguistics, 139
- Incompatible Time Sharing System (ITS), 522
- Incompleteness:
  - metalevel programming, 549
  - knowledge representation, 889
  - theorem, 543
- Inconsistency, Turing machine, 740
- Incremental refinement:
  - cognitive science, 121
  - concept learning, 190–191
- In-Depth Understanding*, 81
- Indexicality:
  - self-reference, 1007–1008
  - of time, 873
- Indexing:
  - frame theory, memory organization, 305–306
  - information retrieval:
    - descriptions, 419
    - techniques, 420–421
- Indirect speech act (ISAs), 143, 1062–1063
  - intention recognition, 1064
- INDUCE program, 191
- Inductive generalization, concept learning, 188–190
- Inductive inference, 409–416
  - applications, 416
  - complexity, 414–415
  - concept learning, 187–188
  - direct methods, 414
  - editing by example facility, 416
  - function classes, 409–411
    - BC criteria, 410–411
    - EX criteria, 410, 415–416
  - grammatical inference, 411
  - identification criteria, 415–416
    - outside limits, 416
    - probably approximately correct, 416
    - reliability, 415
  - language classes, 411–412
  - rules, 189
  - search and its variants, 412–413
    - h-easy* function, 412–413
    - h-honest* function, 413
  - efficiency, 413–414
  - more powerful search, 412–413
- Turing test, 1127
- Inductive logic, 537
- Industrial automation, 35–39
  - adaptive tools, 39
  - artificial intelligence and, 38–39
  - engineering design, 36–37
  - flexibility and quality, 38–39
  - limits, 496
  - load balancing, 38
  - manufacturing operations, 36
  - objectives, 35
  - process planning, 38
  - real-time logistics, 38–39
  - social impact, 36
  - software technology, 38
  - stereo vision, 1083
  - taxonomy, 36–37
- Industrial inspection, template matching, 576
- Industrial programming, automatic programming, 32–33
- Industrial robots (IR):
  - applications, 948–949
    - assembly and arc-welding, 948
    - machine loading-unloading, 948
    - material-handling, 948
    - nonindustrial, 949
  - collision avoidance, 952–953
  - communication with other computer-machines, 946–947
  - control systems, 949–950
  - end effector, 945–946
  - future applications, 954
  - mobile, 961
  - navigation, 953
  - path planning, 952–953
  - planning, 952
  - programming methods, 947–948, 950–951
    - lead-through, 947
    - teach-box, 947
  - self-replication, 1010–1011
  - sensors, 951
    - tactile sensing, 952
  - taxonomy, 945–947
  - teleoperators, 1100–1101
- Inertial navigation systems, (INSs), autonomous vehicles, 41
- Inference, 418–419
  - agenda-based systems, 3–4
  - analogue representation, 881
  - bidirectional processing, 784
  - binary resolution, 892–893
    - rules, 895–898
  - case grammar systems, 335–336
  - causal reasoning, 827
  - commonsense reasoning, 833–834
  - complexity measure, 415
  - conceptual dependency, 198
  - deductive, 187–188
  - default reasoning, 841
  - discourse understanding, 242
  - distance ordering, 425
  - emotion modeling, 274
  - engines, 418, 499
    - propositional networks, 1016
  - exemplary generalization, 419
  - expectation-driven parsing, 696–697
  - Fifth-Generation computers, 179–180
  - general problem solver (GPS), 323
  - goal determination, 675
  - grammatical, 726
  - image understanding, 393
  - inductive, 409–416, 419
  - epistemology, 283

- grammatical inference, 411
- logic, 537
- inexact, medical advice systems, 586
- inheritance, 422
- intelligent computer-aided instruction, 154
- knowledge representation, 886
  - declarative, 886–887
  - validity, 884
- knowledge system techniques, 289
- limits of artificial intelligence, 495
- logical, 418, 536
- epistemology, 281
- nonclassical, 418
- logical programming, 544
- menu-based natural language, 594–595
- meta-knowledge, 598
- military applications, 613
- modus ponens*, 541
- MS. MALAPROP, 632
- natural-language interface, 658
- natural-language processing, 662
- natural-language understanding, 662
- negative information, 419
- NON-VON, 678
- pattern patching, 719
- perception and dot-pattern analysis, 254
- planning action, 749
- plausible, discourse understanding, 239–240
- primitives, 761
- probabilistic or statistical, 419
- programming environment, 790
- PROLOG system, definite-clause grammar, 341
- question answering, 816–818
- reasoning and, 822, 824–825
- rules, propositional logic, 561
- self-reference, 1005–1006
- semantic network, 1012, 1026
- SNePS system, 1049
- speech acts, 1063–1064
- statistical, 419
- story analysis, 1091
  - goal subsumption, 1094–1095
- systems, 418–419
- theorem proving, 1121–1122
- universal generalization, 541
- Inflectional system, word formation, 619–620
- Information:
  - hiding, actor formalism, 2
  - image features, edge detection, 258–259
  - incomplete and inconsistent, problem solving, 247
  - intelligence and, 432, 435–436
  - learning, 438
  - object-oriented programming, 452
  - office automation, 682
  - processing:
    - code analysis approach, 436
    - cognitive-components method, 436
    - cognitive-content method, 436
    - cognitive-correlates method, 435–436
    - cognitive-training method, 436
    - stage analysis approach, 436
    - whole-task approach, 436
  - retrieval, 133, 419–421
  - AI techniques, 420–421
  - computational linguistics, 134
  - constraint satisfaction, 209
  - evaluation of results, 421
  - goal hierarchies and ordering, 755
  - intelligence and, 433
  - law applications, 457
  - LUNAR system, 564
  - request, 419
  - systems, 419–421
  - storage, 452
- Information-processing psychology, *see* Cognitive psychology
- Infrared (IR) proximity sensors, 39
- Infrared spectroscopy:
  - chemical instrumentation, 95
  - chemical structure elucidation, 94
- INGEST system, computational linguistics, 136
- Inheritance:
  - concept learning, 187
  - frame-based, 422
  - frame theory, 308
  - multiple frame theory, 307
  - object-oriented programming
    - knowledge sharing, 454–455
    - medical example, 453–454
  - property, semantic memory, 593
  - semantic networks, 1017–1018
  - shortest-path, 424, 426
  - see also* Property inheritance
- Inheritance hierarchies, 422–430
  - artificial intelligence, 12
  - demons, 429
  - first-order logic, 423–424
  - inheritable relations, 428–429
  - logic programming, 556
  - multiple inheritance, 422–423
    - exceptions, 424–426
  - nonmonotonic reasoning, 849–850
  - object-oriented programming, 556–557
  - parallel-marker propagation, 429
  - programming languages, 429–430
  - properties and slots, 422
  - simple exceptions, 424
  - slot constraints, 426
  - splits and partitioning, 427–428
  - structured concepts, 426–428
  - tangled, 422–423
  - taxonomy, 422
- Input-output specifications, stating the problem, 19–20
- INSPECTOR system, 529
- Instance variables, object-oriented programming, 452, 454–455
- Instantiation, Herbrand
  - binary resolution, 893
  - theorem proving, 1120
- Instant Insanity game, constraint satisfaction, 209
- Institute for Theoretical Experimental Physics (ITEP), 159–160
- Instruction sets, computer systems, 176
- Integrated Diagnostics projects, military applications, 611
- Integrated maintenance information system (IMIS), 611
- Integrated telephone systems, 682
- Integration:
  - image understanding, 389–390
  - multisensor, 632–637
  - programming assistants, 786
  - solution path, 474–475
- Integration Partial Parser (IPP), *see* IPP
- INTELLECT system, 141, 350, 431
  - computational linguistics, 139
- Intelligence, 431–439
  - defined, 432–433
  - development, 436–437
    - formalized structures, 437
  - expert systems, 287–288
  - global theories, 432–433
  - learning, 436–438
    - environment, 436
    - inconsistent information-processing components, 438
    - information processing, 438
    - psychological aspects, 437
    - transfer aspects, 437–438
  - psychological and social issues of AI, 1057–1058
  - societal influences, 432
  - structure of, 433–437
    - circle theory, 435
    - facet and hierarchical theories, 433–435
    - fluid intelligence-crystallized intelligence distinction, 435
    - g ability factor, 434–435
    - general intelligence and primary mental abilities, 433
    - information processing, 435–436
    - spatial ability, 435
  - tests, 432
- Intelligent Automatic Test Generation (IATG), 610
- Intelligent communications interfaces (ICI), 44
- Intelligent computer-aided instruction (ICAI), 154–159, 719
  - accessibility, 157
  - components, 267–268
  - computer-aided instruction, 157
  - design issues, 156–157
  - education applications, 267
  - future trends, 158–159
  - human-computer interaction, 384–385
  - implementation issues, 157–158
  - medical advice systems, 587
  - natural-language generation, 642–654
  - performance factors, 157–158
  - program examples, 154–155
- Intelligent execution of logic programming, 554–555
- Intelligent-user interface:
  - Fifth-Generation computers, 179–180
  - human-computer interaction, 385
- Intensity changes:
  - edge detection, 257–266
    - directional or nondirectional operators, 264
  - multiple scales, 264–265
  - one-dimensional detection, 257–261
  - recovering physical world properties, 266
  - smoothing and differentiation, 258–261
  - spatial filters, 264
  - two-dimensional detection, 261–262
- Intensity image processing:
  - robotics sensing, 932–933

- complete outline, 932
- gray-level image, 933
- partial or connected outline, 933
- Intentionality, phenomenology, 731, 736–737
- Interaction:
  - cooperative, distributed problem-solving, 246
  - discourse understanding, 240
  - human-computer, 383–385
- Interest operator, path planning and obstacle avoidance, 711
- Interfaces:
  - natural-language, 655–659
    - conjunction and disjunction, 659
    - ellipsis, 658
    - end-user control of interpretation, 658
    - English as appropriate language, 656
    - inference, 658
    - negation, 658
    - pronoun use, 658
    - prospective users, 657–659
      - coverage and habitability, 657–658
    - quantification, 658
    - state of the art, 656–657
    - telegraphic input, 659
    - time and tense, 658–659
    - ungrammatical input, 659
- INTERLISP, 510, 523
  - computer architecture, 216
  - history: 1960–1970, 522
  - human-computer interaction, 385
  - MASTERSCOPE and, 794–795
  - Programmer's Assistant, 785
  - programming environment, 789, 794–795
    - CLEANUP function, 795
    - file package, 794–795
    - interactive interface, 794
- InterLisp-10, 524
- InterLisp-D, 524
- International Computer Chess Association, 160
- International Joint Conference on Artificial Intelligence (IJCAI), 16, 461
- INTERNIST system, 82, 394–395, 440
  - development of, 587–588
  - disease frames, 585
  - domain knowledge, 252
  - evoking strength, 587
  - frequency weight, 587
  - knowledge engineering, 288
  - theoretical basis, 585–586
- INTERNIST-I, causal reasoning, 828
- Interpretation:
  - context of:
    - discourse understanding, 237
  - discourse understanding:
    - context-dependence, 238
  - distributed, 246
  - frame theory, 304
  - hermeneutics, 364–365
  - LISP, 518
  - natural-language interface, 658
  - truth, 540
- Interpretation-guided segmentation (IGS) program, 394–395
- Interstellar navigation, mobile robots, 961
- Intervals, temporal reasoning, 871
- Interval tree, scale-space imaging, 977–978
- Intrinsic images:
  - depth representation, 1153
  - early vision, 1142
- Introspection, self-reference, 1008–1009
- Inverse filtering, texture analysis, 1105
- Inverse optics, early vision, 1141
- IPP (integrated partial parser):
  - discourse understanding, 240
  - event builders, 699
  - event refiners, 699
  - expectation-driven parsing, 699–701
  - function words, 699
  - memory organization packets, 592
  - natural-language understanding, 663
  - script acquisition, 992
  - token makers, 699
  - token refiners, 699
  - world knowledge acquisition, 1096–1097
- IQ, creativity and, 224
- IRLIST, 533
- IRUS, natural-language interface, 657
- ISA hierarchies:
  - artificial intelligence, 12
  - frame theory, 306–309
  - goal-directed activation, 395–396
  - inheritance:
    - redundant, 422, 425
- Isaacson, L., 639
  - problem solving, 775–776
- ISing model ferromagnetism, 1140
- ISIS system:
  - computer-integrated manufacturing, 173
  - planning, 756
- Iterative deepening:
  - A\* (IDA\*), 998
  - Chess 4.5, 99
  - computer chess, 164
    - software advances, 166–167
  - depth-first, 996
- Iterative search:
  - computer chess
    - software advances, 166–167
- ITP program, binary resolution, 900
- Jacob, R., 383–387
- James, W., 115
- Japanese Robotic Industries Association (JRIA), 945
- JASON system:
  - autonomous vehicles, 39
  - dead reckoning position location, 42
  - route search, 43
- Jensen, A., 432, 436
- Jet Propulsion Laboratory (JPL) Rover, 39
  - dead reckoning position location, 42
  - obstacle avoidance, 42
  - route search, 43
- JETS system, database interface, 138
- Joblessness, AI and, 1053–1055
- Job shop scheduling, beam search, 56
- JOHNNIAC computer, LISP development, 519
- Joint transformation, robot-control systems, 910
- Jones, K. Sparck, 419–421
- Jones, M., 133–146
- Journal of Philosophical Logic*, 537
- Journals of artificial intelligence, 531, 533
- JUDGE system, 592
- Judicial decisions and artificial intelligence, 457
- JUDITH program, 457–458, 460
- JUMP arcs, ATN grammar, 329, 331
- Jurisprudence, *see* Law applications
- Justification, epistemology, 280–281
- Justification-based belief systems, 60–61
- KAISSA program, 159–160, 441
- KAMP system, text generation, 145
- Kanade, T., 130
  - control structure, 395
- Kanal, L. N., 720–728
- Kant, Immanuel, 282
- Kaplan, Julio, 735
- KAS system, domain knowledge, 252
- Kay, Martin, machine translation system, 570
- KDS system, text generation, 145
- KEAL system, speech understanding, 1081
- Kearsley, G., 154–159
- KEE system:
  - domain knowledge, 253
  - expert systems, 95
- Kelley, K. L., 443
- Kelvin stability theory, 915
- Kempelen, Baron von, 159
- Keough, R., 890–892
- Kepler's Third Law of Planetary Motion, 485
- Kernel sentences:
  - game playing, 316
  - STUDENT system, 1099
- Keyser, S. J., 744–746
- Keyword indexing, Chemical Abstracts Service, 94
- Kieras, D., 111–114
- Killer heuristic, 163
  - computer chess methods
    - software advances, 166–167
- Kinematics, robot-control systems, 910, 925
- Kirchoff's law, 810, 836
- KL-ONE system, 66, 309, 441
  - case grammar, 335
  - frame theory, 309
  - inheritance hierarchies, 422, 426–427
    - arches, 426–428
    - demons, 429
  - knowledge representation, 888–889
  - propositional networks, 1016
  - semantic networks, 1019
    - inheritance hierarchy, 1017–1018
  - SUPERC relation, 422
- KL-TWO, semantic networks, 1020
- k*-means procedure, pattern recognition, 723
- k*-nearest-neighbor density-estimation procedure, 723
- KNOBS (Knowledge-based System), 609
  - planning, 756
- Knowing-not phenomenon, meta-knowledge, 599
- Knowledge:
  - analysis, 740
  - belief systems, 63
  - epistemology and, 280
  - procedural, 9
  - sources of, 280



- Knowledge acquisition:  
 cognitive science, 120–121  
 emotion modeling, 274  
 knowledge system construction, 294  
 medical advice systems, 585–587
- Knowledge-based systems:  
 art in AI, 8–9  
 Bayesian decision methods, 55  
 belief revision, 58, 61  
   context-layered bases, 58  
 blackboard architecture, 73  
 cognitive modeling, 113  
 competition principle, 848  
 computer-integrated manufacturing, 173  
 construction, 294–295  
 dynamic, 816  
 edge detection, early vision, 1137–1138  
 efficiency principle, 849  
 Fifth-Generation computers, 179–180  
 goal satisfaction principle, 849  
 hermeneutics, 372  
 industrial robots, 949–950  
 inheritance, 422  
 intensional, 816  
 knowledge representation, 882–883  
 Levesque theories, 67  
 limits of, 500  
 machine translation, 567–568  
 maintenance and acquisition, 599  
 meta-knowledge, 598–599  
 military applications, 607–608, 610  
 NON-VON systems, 678  
 organization and design, 292–293  
 problem solving, 770  
 programming assistants, 786–787  
 programming environment, 790  
 rule-based systems, 965  
 semantics, 1028–1029  
   advantages and disadvantages, 1028–1029  
   procedural semantics and, 1029  
 significance principle, 849  
 techniques, 288–291  
 tools for building, 295–298  
 validity principle, 849
- Knowledge-based vision, 389
- Knowledge dependency graph, story analysis, 1097–1098
- Knowledge engineering:  
 case-based reasoning, 477–478  
 current status of tools, 298  
 domain knowledge, 252  
 expert systems, 288  
 frame theory, 305–306  
 fundamentals, 291–294  
   beliefs, 291–292  
   facts, 291–292  
   heuristics, 291–292  
   knowledge system organization and design, 291–293  
 history, 288  
 knowledge-processing tasks, 294–295  
 knowledge system architecture, 293  
 medical advice systems, 587  
*see also* Expert systems
- Knowledge interactions, computer chess, 169
- Knowledge networks, ICAI programs, 154
- Knowledge representation, 882–889  
 automatic programming, 19  
 belief systems, 63  
 biological systems, 924  
 causal reasoning, cognitive mental models, 831–832  
 commonsense reasoning, 833–834  
 completeness, 132  
 computational linguistics, 143  
 computer systems, 174–175  
 conceptual dependency, 194  
 connectionism, 200  
   model, 201–202  
 control structures, 887  
 current research, 889  
 declarative approach, 600  
 dictionary viterbi algorithm, 86  
 education applications, 267  
 epistemology, 284–285  
 exact representation, 86  
 frame theory, 885  
 FRL system, 882–883, 888  
 game-playing programs, 88  
 history, 882–883  
 image understanding, 398  
 imprecision, 889  
 inference, 886  
   declarative, 886–887  
 KL-ONE system, 441, 888–889  
 knowledge-base semantics, 1028–1029  
 KRL system, 441, 882–883, 888  
 limits of, 500  
 logic and, 537  
 logic databases, 551  
 matching, 886  
 medical advice, 584–585  
 memory organization packets, 591–592  
 meta-knowledge, 598  
   connecting theories, 601  
   natural-language understanding, 662  
   nonmonotonicity and defaults, 889  
   office automation, 682  
   organization of knowledge, 885–887  
   path planning with static knowledge, 709  
 PLANNER system, 887–888  
 primitives, 759–761  
 problem solving, 770  
 programming environment, 792  
 procedural and declarative, 884–885  
 production systems, 888  
 PROLOG system, 887  
 reasoning, inference, 826  
 SCHOLAR system, 980  
 self-reference, 1006  
 semantics, 593–594, 883–885, 1026  
 social issues of AI, 1050–1051  
 speech understanding, 1081  
 techniques, 288  
 trie, 86  
 vocabulary, 882
- Knowledge representation language (KRL), 308, 441  
 agendas in, 4, 848  
 frame theory, 306–307  
 hermeneutics, 371  
 inheritance hierarchies, 422  
*see also* Epistemology, Frame theory, Inheritance hierarchy, Metaknowledge, metarules, and metareasoning, Representation, knowledge, and Semantic networks,
- operations and procedures, 882–883, 888  
 pattern matching, 719  
 semantic networks, 1013
- Knowledge sharing, object-oriented programming, 454
- Knowledge sources:  
 AGE, 78  
 BB1, 78–79  
 blackboard architecture, 74–76, 79  
 character recognition, 84–85  
 HASP system, 76  
 HEARSAY-III, 78  
 HEARSAY-II, 75  
 OPM system, 77  
 reasoning, 848  
 speech understanding, 1079–1080
- Knuth-Bendix completion algorithm, 1120–1121
- Kobsa, belief systems, 66
- Kodak Wratten filters, 126
- Koditschek, D. E., 892–921
- Koffman, E. B., 315, 317
- Kolodner, J., memory organization packets, 592
- Konolige, K.  
 belief systems, 66  
 heuristic theories, 66–67
- Korf, R. E., 376–380, 994–998
- Koskenniemi, K., 619–620
- Kosslyn, E., 117
- Kowalski, R. A., 544–557
- Kramer, B. M., 882–889
- Kriegspiel chess, minimax procedures, 614–615
- Kripke models, modal logic, 618
- KRL-O language, primitives, 760
- Krumme network, feature analysis, 83–84
- KRYPTON system:  
 domain knowledge, 253  
 frame language, 309  
 knowledge representation, 883, 889  
 semantic networks, 1019–1020
- K-tails inductive inference method, 414
- Kuhn, T. S., scientific hermeneutics, 370
- Kuiper's algorithm, 866
- Kuipers, B., 827–832
- Kulikowski, C., 252–253
- Kurzweil Corporation, 86
- Kurzweil reading machine, 800
- Kyllonen, P., 436
- LABELS construct, LISP, 515
- Lack-of-knowledge inference, 599
- LADDER system, 488  
 case frame ellipsis resolution, 673  
 database interface, 138  
 natural-language understanding, 662, 667
- Lagrangian formulation:  
 obstacle avoidance, 42  
 robot-control systems, 909–911  
 robotics manipulators, 926
- Laing, R., 1010–1011
- Lambda calculus, 441–442  
 applications to artificial intelligence, 442  
 basics, 442  
 definability, 442  
 foundations, 442  
 notation, 442
- Lambda expression:  
 InterLisp, 524

- LISP system, 511, 520
  - variables, 511–512
- lists, 515–516
- Landmark recognition:
  - autonomous vehicles, 41
  - mobile robots, 958–959
- LANDSAT system:
  - cluster analysis, 105
  - texture modeling, 1108
- Langley, D., 447
- Langley, P., 464–485
- Language:
  - case frames, 333–338
  - cognitive psychology, 118
  - cognitive science, 121–122
  - communication and, 838
  - computer systems, 174–175
  - cybernetics of, 226–227
  - deficits in cognitive science, 121
  - discourse understanding:
    - dialogue *vs.* text research, 243–244
    - pragmatic perspective, 241
  - domain knowledge, 252–253
  - form and function disparity, 505
  - frame theory, 303–304
    - ISA hierarchies, 308
    - KL-ONE, 441
    - KRL, 441
  - organizational features, 308–310
  - game playing:
    - hermeneutics and, 369–370
    - importance of, 317–318
  - generation and conceptual dependency, 198–199
  - hermeneutics and, 366, 369–370
  - impairment and prostheses, 803–804
  - inductive inference, 411–412
    - positive *vs.* complete representations, 411–412
  - k-reversible regular, 414
  - object-oriented, 452–456
    - frame theory, 309–310
  - parallelism in DADO, 228
  - predicate logic, 538–541
  - primitives, 760–761
  - processing:
    - competence-performance dichotomy, 506–507
    - social issues of AI, 1051
  - programming environment, 791–792
  - programming, limits of, 500–502
  - propositional logic, 559–561
  - semantic grammar, 351
  - speech and, 503
  - structure and speech synthesis, 1070
  - transformational grammar, 354
  - understanding:
    - case frames, 334
      - obligatory slots, 334
    - semantic memory, 593
  - see also* Context-free language; Program-  
ming language; specific languages  
and grammars and Natural lan-  
guage headings
- Language acquisition, 145–146, 443–450
  - cognitive models, 446–450
    - AMBER model, 447–450
    - CHILD model, 447–450
  - grammar systems, 448–449
  - Hill's model, 447–448
  - learning, making and correcting errors, 449–450
  - McWhinney's competition model, 448–450
    - psycholinguistic studies, 446–447
  - competence-performance dichotomy, 506
  - historical notes, 443
  - innatists *vs.* empiricists, 443–444
  - intelligence and, 437
  - machine learning, 465, 479–481
  - present models and future trends, 450
  - taxonomy, 443–444
  - theoretical issues, 444–446
    - connectionist models, 446
    - learnability models, 445–446
  - transformational grammar, 353–354
- Language Acquisition System (LAS), 443
  - cognitive modeling, 113
  - language acquisition, 146
  - sentence-meaning pairs, 480–481
- Language Craft, 663
- Language-delaying handicap, 140
- Language-learning program, 443
- Laplacian equation:
  - early vision, 1133–1134
  - intensity change detection, 263
  - texture analysis, 1104
- Large processing elements (LPEs), 678
- Larkin, K. M., 304
- Lateral geniculate nucleus (LGN), 1136
- Lateral inhibition:
  - connectionism, 202
  - early vision, 1133–1134
- Lattices:
  - frame theory hierarchies, 307
  - inheritance hierarchies, 423
  - semantic networks, 1017
- Law applications of artificial intelligence, 456–461
  - background, 457–458
  - cognitive processes, 461
  - legal analysis programs, 458–459
    - Gardner's dissertation, 459
  - tasks for, 460
- LAWGICAL program, 460
- Lazy evaluation, coroutining, 219, 222
- Lazy execution, 555
- Learning:
  - alienated *vs.* syntonic, 184
  - analytical, 472
  - artificial intelligence and, 14–15
  - automated programming, 18
  - autonomous vehicles, 43–44
  - beam search, 56
  - belief revision, 61
  - Boltzmann machine, 81
  - checkers-playing programs, 90–92
  - chemistry, 93
  - cognitive modeling, 113
  - cognitive psychology, 116
  - cognitive science, 122
  - commonsense reasoning, 833
  - computers in education, 182–183
  - concept, 185–192
    - analogous, 186
    - classical view, 192
    - data-driven, 190
  - deductive, 186
  - defined, 185
  - exemplar view, 192
  - extended notions, 191–192
  - incremental specialization, 190–191
  - inductive inference, 186–187, 190
    - hypothesis generation, 187–188
    - rules, 189
  - instance space *vs.* description space, 189–190
  - instructive, 186
  - knowledge implantation, 186
  - mixed methods, 191
  - model-driven, 190–191
  - observation and discovery, 186–187
  - preference criterion, 187–188
  - premise statements, 188
  - probabilistic view, 192
  - typicality, 192
  - validation, 191
- connectionism, 200, 202–203
  - network construction, 204
- demons, 232
- education applications of AI, 181, 268–271
- environments for, 184–185
- epistemology, 284
- EURISKO program, 287
- evaluation functions, 379
- explanation-based, 472
- game-playing programs, 316–318
  - checkers, 88
- inductive inference, 409
- intelligence, 436–438
  - inconsistent information-processing components, 438
  - information processing, 438
  - transfer aspects, 437–438
- intelligent computer-aided instruction, 154
  - strategies, 158
- language acquisition in cognitive models, 449–450
- limits of artificial intelligence, 500
- machine, 464–485
  - by analogy, 476–479
    - case-based reasoning, 477–478
  - derivational-analogy method, 477–478
  - generalized plans, 478–479
- cognitive modeling, 111
- concepts from examples, 465–475
  - analytic approaches to concepts learn-  
ing, 471–473
  - Aq algorithm, 471
  - arch concept, 465–466
  - combined approaches, 469–470
  - decision tree construction, 470–471
  - empirical learning, 473
  - general-to-specific methods, 468–469
  - open problems, 473
  - search, 466–467
  - specific-to-general methods, 467–469
- by discovery, 481–485
  - description and explanation, 485
  - qualitative laws, 483–484
  - quantitative laws, 484–485
  - taxonomy formation and conceptual  
clustering, 481–483
- history, 464–465

- language acquisition, 479–481
- problem solving, 473–477, 777
  - assigning credit and blame, 474–475
  - heuristics, 473–476
  - open problems, 476
  - learning-from-solution-paths, 475
  - learning-while-doing, 475
- semantic networks, 1021–1022
- macro operators, 583
- memory and, 117
- memory organization packets (MOPs), 592
- natural-language interfaces, 655
- office automation, 681–682
- perceptrons, 729–730
- problem solving, 767
- robot-control systems, 919–920
- script acquisition, 991–992
- semantic networks, 1012
- Turing machines, 1125
- unsupervised, 187
  - pattern recognition, 721
- world knowledge acquisition, 1096
- Learning Research Development Center, 184
- Least-squared error methods:
  - Hough transform, 383
  - motion analysis, 629
- Lebowitz, M., 592
- Left-corner parsing, 690–691
  - algorithm implementation, 694
- Legal analysis programs, 458–459
  - open texture, 459
- Legal documents and artificial intelligence, 460
- Legged locomotion, 957–958
- LEGOL project, 460
- Lehnert, W. G., 273–274, 371, 1090–1098
- Leifer, L., 797–804
- Lenat, D., 7
- Lesgold, A., 267–271
- Lesser, V., 245–250, 362
- Let expression:
  - LISP 1.5, 522
  - LISP system, 511
    - compilation, 519
    - variables, 511–512
  - lists, 515
- Levesque, H., 308
  - belief systems, 67
  - speech act theory, 68
- Levi, E., 457
- LEX project in problem solving, 777–778
- Lexical analyzer, 145
  - coroutines, 219
  - natural-language generation, 649, 657
- Lexical functional grammar (LFG), 359
  - ATNs, 329–300
  - competence and performance, 507–508
  - transformational grammar, 354
- Lexical-interaction language (LIL), 703–704
- Lexical lookup in cognitive science, 121
- Lexical variables:
  - LISP, 511
- Lexicon:
  - frame theory, 305
  - morphology, 619
- Lexor function in MacLisp, 522–523
- Lieberman, H., 452–456
- Liebniz, Gottfried Wilhelm, 282
- LIFER system, 488
  - case frame ellipsis resolution, 673–674
  - database interface, 138
  - natural-language understanding, 663, 667
  - semantic grammar, 350, 352–353
- Light-emitting diode (LED) sensor:
  - industrial robots, 946
  - proximity sensing, 805
- Likelihood ratio, 846
  - Bayesian decision methods, 49
- Limits of artificial intelligence, 488–501
  - brain as biochemical computer, 488–489
  - brain's "programs", 490–491
  - computational complexity, 491–492
  - moral limits, 502
  - philosophical issues, 488
  - present state of knowledge, 492–501
    - AI methodology, 500–501
    - motor control, modeling of spatial environments and motion planning, 496–497
    - reasoning, planning, knowledge representation and expert systems, 497–501
    - sensory functions, 492–496
- programming languages, 500–502
  - automatic, 501–502
- quantitative estimates of brain's computing power, 489–490
- Linear algorithm:
  - motion analysis, 622
  - planar patches, 628
  - optical flow, 626
- Linear operators and feature extraction, 300–301
- Linear polynomial program in checkers-playing, 91
- Linear precedence rules for GPSG, 343
- Linear predictive coding (LPC)
  - phonetic segments, 1075
  - speech synthesis, 1071
- Linear resolution and theorem proving, 1118
- Linear-time-invariant systems for robot control, 906–908
  - gain scheduling, 919
- Linear transformations in color vision, 127–128
- Line detection, connection machines, 200, 204
- Linguistic case grammar, frame theory, 308
- Linguistic string project, 142
- Linguistics:
  - augmented transition network grammar, 324
    - experimentation, 328–329
  - cognitive psychology, 115
  - competence and performance, 503–508
    - dichotomy, 503–507
    - evidence of dichotomy, 505–507
    - utilization of, 507–508
  - computational, 133–146
  - conceptual dependency, 194
  - generalized phrase structure grammar, 342–343
  - language acquisition, 444–446
  - machine translation, 566
- natural-language processing, 661
  - phonemes, 744–746
  - semantics and, 1025
  - Turing test, 1129
- Linking, planning and, 752
- Lipovski, G. J., 18
- Liquids, physics of, 836
- LISCOM, 523
- LISP (List processing), 508–527
  - artificial intelligence language, 15
  - augmented transition network (ATN)
    - grammar, 328
    - misconceptions, 330–331
  - automatic programming synthesis, 22, 25–26, 29
    - function-merging technique, 26
    - recurrence relations, 26
  - backquote, 514
  - basics, 509
  - bindings, 511–512
    - deep-binding, 512
    - shallow-binding, 512
    - special bindings, 511–512
  - chemical structure, 95
  - Church's thesis, 99–100
  - closures, 515
  - cognitive language acquisition models, 448
  - cognitive modeling, 113
  - comparative history:
    - 1956–1960, 519–522
    - implementation, 520–521
    - READ AND PRINT programs, 520
    - 1960–1970, 522
    - 1970–1980, 522–525
    - 1970–1985, 525–526
    - 1980–present, 526
  - compilation, 518–519
    - separate, 526–527
  - computer systems, 174–175
    - applications, 177
    - architecture, 215
    - storage management, 175
  - COND and, 509, 516
    - connection machine usage, 177–179
  - constraint satisfaction, 210
  - control structures, 212–213
    - programming language design, 212
  - coroutines, 219
    - data abstraction, 514
  - debugging environment, 527
  - declaring variables special, 512
  - dialects of, 524
  - Drilling Advisor system, 291
  - extending, 526
  - fixnums, 510–511
  - free and bound references, 512
  - functions, 513
    - anonymous, 513
  - GPSG parsers, 343
  - human-computer interaction, 385–386
  - inductive inference, 410
    - direct methods, 414
    - language acquisition, 445
  - industrial robots, 950–951
  - intelligent computer-aided instruction (ICAI)
    - design issues, 157
    - performance, 157–158

- interpretation, 509, 518
- I-rules, 668
- knowledge engineering, 296
- knowledge representation, 883–884
- lambda calculus, 441–442
- language:
  - extension, 455–456
  - inheritance hierarchies, 422
  - limits of, 500
- legal analysis programs, 459
- macros, 513–514
- object-oriented programming, 455
- parsing and empirical speed comparisons, 694
- pattern matching, 716
- program development strengths, 526–527
- programming environment, 790, 792
- prostheses, 799
- read-eval-print loop, 512
- real implementation, 527
- recursion, 875
- run time typing, 517–518
  - data structures, 518
    - characters and strings, 518
    - numbers, 518
    - vectors and arrays, 518
- scheme, 525
- self-reference and introspection, 1008
- semantic networks, implementation, 1022
- S-expression, 510–511
- SHRDLU and, 1048
- simple evaluation model, 516–517
  - AND, 516–517
  - COND, 516
  - PROG and GO, 517
- simple example, 508–509
- SMALLTALK and, 1049
- social issues of AI, 1050
- symbols, identifiers, locations and bindings, 511–513
- syntax, 509–510
  - CONS cells and lists, 510
  - manipulating list structure, 510–511
    - algebraic syntax, 510–511
    - common LISP, 510–511
    - programs and data, 510
  - symbols and atoms, 509
- theorem proving, 1122
- typelessness, 527
- value cell, 511
- word-expert parsing, 703
- LISP 1, 520–521
- LISP 1.5, 521–522
  - special and common variables, 521–522
- Lisp 1.5 Programmer's Manual*, 521
- LISP 1.6, 524
- LISP Machine, 528–529
  - AREA feature, 529
  - commercial, 525–526
  - development, 525–526
  - flavors, 309
    - inheritance hierarchies, 422, 429–430
  - garbage collection, 529
  - graphical display console, 529
  - large virtual-address space, 529
  - LISP dialect, 525, 529
  - programming tools, 794
  - software, 526, 529
- LISP Machine, Incorporated (LMI), 525–526, 529
- LISP 360, 525
- LISP 370, 512, 525
- LISP/VM, 512, 525
- Listening, musical, AI and, 638
- Literature for artificial intelligence, 530–534
  - anti-AI literature, 532
  - beginnings, 530
  - cognitive science, 531–532
  - expert systems, 533
  - fifth generation computers, 533
  - future trends, 533–534
  - historical literature, 532
  - LISP literature, 532
  - for logic, 537–538
  - logic programming and PROLOG, 532–533
    - 1960s, 531
    - 1970s, 531–532
    - 1980s, 532–533
    - overview, 532
  - pattern recognition, 720–721
  - pre-AI literature: 1940s, 530
  - qualitative physics, 813
  - robot-arm-control, 902
  - social issues of AI, 1059
  - speech understanding, 1081
- Litigation support system, artificial intelligence, 460
- Llewellyn, K., 457
- LMA program:
  - binary resolution, 900
  - reasoning and, 823
- LNR system, cognitive modeling, 113
- Locality principles in transformational grammar, 358
- Localization assumption and intensity changes, 265–266
- Local operators, feature extraction, 300–301
- Locations:
  - conceptual dependency rules, 195–196
  - connectionism, 202
  - LISP system, 511
- Locke, John, 282
- Lockhart, R., 117
- Loftus, E., 204
- Logic, 536–538
  - artificial intelligence and, 12–13, 537
  - bibliography of articles, 537–538
  - chemistry, 93
  - Church's thesis, 99
  - clausal form, 545
  - clustering, implicative rules, 110
  - completeness, 131
  - default reasoning, 842–843
  - definite-clause grammar, 339–340, 342
  - discourse understanding, 239
  - epistemological, belief system, 64–65
  - first-order, *see* First-order logic
  - formal in law applications, 457
  - inheritance, 423–424
  - intensional, semantics, 1026
  - knowledge system construction, 294
  - machine translation, 565
  - modal, 617–619
    - dynamic logic, 619
    - first-order, 618–619
  - formalisms, 617–618
    - axiomatic systems, 618
  - inference, 418
  - Kripke models, 618
  - temporal logic, 618
- nature of, 536–537
- neat/scruffy debate, 537
- positivism:
  - empiricism and, 742
  - hermeneutics, 369–370
- predicate, 538–543
  - clustering, 108
  - deductive systems, 541
    - axiomatic, 541
  - extensions, 541–543
  - identity, 541
    - descriptions, 541–543
  - frame theory, 308–309
  - inference, 418
  - introduction and elimination rules, 541–542
- language, 538–540
  - expressibility, 540–541
  - semantics, 540
  - syntax, 539
  - metatheoretic results, 543
  - second-order logic, 543
- procedural semantics, 1030
- propositional, 558–563
  - artificial intelligence, 562
  - conjunctive normal form (CNF), 560
  - deductive systems, 561–563
    - semantics, 562–563
    - syntax, 561–562
  - disjunctive normal form (DNF), 560
  - equivalences, 560
  - inference, 418
  - language, 559–561
    - paradox of the material conditional, 561
  - semantics, 560–561
    - connectives, 560
  - syntax, 559–560
  - tautologies, contradictions and contingent propositions, 561
  - two-place truth-functional connectives, 559–561
- semantic networks, 1018–1019
- systems of, 537
- template representation, 459
- temporal, 618
- theorem proving, 1115–1116
- thinking and, 118
- see also* specific types of logic
- Logic programming, 544–557
  - binary resolution, 893
  - constraint satisfaction, 210
  - control structure, 215
  - coroutines, 222
  - DADO computer, 228
  - databases, 551–552
    - query optimization, 552
    - relational, 551–552
  - definite-clause grammar, 339–340
  - domain knowledge, 252
  - forward *vs.* backward reasoning, 545
  - frame theory, 309
  - functional programming, 550–551
  - historical origins, 546

- Horn-clause:
  - AND/OR nondeterminism, 546
  - extensions, 548–549
    - conditional subgoals, 549
    - metalevel subgoals, 549
    - negations, 548–549
  - procedural interpretation, 546
- inference, 418
  - nonclassical, 418
- intelligent execution strategies, 554–555
  - loop detection, 555
  - subgoal selection, 554–555
- literature, 532–533
- LUSH corouting *vs.* PROLOG strategy, 546–547
- negation-as-failure rules, 823
- object-oriented programming, 556–557
  - abstract data interpretation, 556–557
  - process data interpretation, 557
- parallelism, 555–556
- problem reduction, 763–764
- PROLOG, 547–548, 796–797
- recursive data structures, 547
  - minimal model semantics, 547
- resolution, 545–546
- rule-based expert systems, 552–554
  - declarative input-output, 553
  - heuristic *vs.* algorithmic programming, 553–554
- self-reference, 1007
- semantic networks, 1011
- specifications, 549–550
- theorem-proving, 545
- Logic Theorist program, 537
  - problem reduction, 766
  - problem solving, 778
  - processing, bottom-up and top-down, 780
  - propositional logic, 562
- Logistics:
  - manufacturing, 37
- LOGLISP language, 551
  - search parallelism, 556
- LOGO language, 563
  - computers in education and, 185
  - education applications, 269
  - recursion, 875
- Lohman, D., 435
- London School of functionalism, 653
- Look-ahead technique in heuristics, 378
- Loop detection, logic programming, 555
- LOOPS system, 564
  - belief revision, 58
  - domain knowledge, 253
  - education applications of AI, 269
  - human-computer interaction, 385
  - inheritance hierarchies, 422, 429
  - LISP machines, 526
  - logic programming, 556–557
- Loveland's model elimination, 546
- Lowerre, B., HARPY program, 362
- LPARSIFAL program, 479
- LPS (logic programming system) for DADO architecture, 228
- LSI system, computer-aided design, 153
- LSNLIS system, procedural semantics, 1027–1028
- Lucas, J. R., 739–740
- LUNAR system, 564
  - augmented transition network grammar, 325–326
  - computational linguistics, 136–137
  - deep structure, 231
  - grammar, 143
  - parsing
    - ATN grammar, 330
    - speed, 694–695
  - procedural semantics, 1027–1028
  - semantics, 1025
- Lunneborg, C., 436
- LUSH resolution, 546
  - logic programming, 546–547
- Lyapunov function, control theory, 905
- Lytinen, J., 592
- McCarthy, John, 508
  - automata theory, 530
  - belief systems, 63
  - circumscription, 100–102, 843
  - computer chess methods, 159–160
  - epistemology, 284
  - LISP development, 519–522
  - neat/scruffy debate, 537
  - situational calculus, 3
- McCarty, L. T., 458–459
- McClelland, J., 232, 446
- McCord, M., 339
- McCorduck, Pamela, 532
- McCulloch, N., 226
- McCulloch, W., 204
- McCulloch-Pitts nerve net, 1125
- McDermott, D. V., 863–874
  - CONNIVER system, 205
- McDonald, D. D., 642–654
  - surface structure, 652
- MACHACK 6, 564
- MACHACK system, computer chess programs, 160
- Mach bands, early vision, 1131, 1133–1134
- Machine Intelligence Corporation, 933
- Machine learning, 464–485
  - artificial intelligence, 15
  - by analogy, 476–479
    - case-based reasoning, 477–478
    - derivational-analogy method, 477–478
    - generalized plans, 478–479
  - concepts learning from examples, 185, 465–475
    - analytic approaches to concept learning, 471–473
    - Aq algorithm, 471
    - arch concept, 465–466
    - combined approaches, 469–470
    - decision tree construction, 470–471
    - empirical learning, 473
    - general-to-specific methods, 468–469
    - open problems, 473
    - search, 466–467
    - specific-to-general methods, 467–469
  - by discovery, 481–485
    - description and explanation, 485
    - qualitative laws, 483–484
    - quantitative laws, 484–485
    - taxonomy formation and conceptual clustering, 481–483
  - empiricism, 742
- history, 9, 464–465
- language acquisition, 479–481
  - negative instances in grammar learning, 481
- PANDEMONIUM system, 687
- phenomenology, 741
- problem solving, 473–476, 777
  - assigning credit and blame, 474–475
  - heuristics, 473–474
    - instances to, 475–476
    - open problems, 476
  - instances to heuristics, 475–476
  - learning-from-solution-paths, 475
  - learning-while-doing, 475
- prostheses, 797
- semantic networks, 1021–1022
- Machine translation, 564–570
  - artificial intelligence, 564–566, 568–570
  - computational linguistics, 133–134, 137, 564–565
  - definite-clause grammar, 341–342
  - expectation-driven parsing, 700
  - history, 565–566
  - knowledge-based systems, 567–568
  - natural-language analysis, 664
  - problems and theories, 565
  - re-emergence, 141
  - semantic memory, 593
  - semantic networks, 1011
  - theoretical basis, 566–567
- Mackworth, A. K., 395, 205–210
- MacLisp, 510, 522
  - CommonLisp, 526
  - computer chess, 168
  - early development, 522
  - later development, 523–524
  - numbers, 521
- Macro facility, 513–514
  - compiler, 794
  - defmacro, 514
  - planning, 748–749
    - hierarchical, 754–755
  - problem solving, 775
  - table:
    - macro operator, 582
    - means-ends analysis, MPS program, 582–583
- Macromolecular structure determination, 96
- MACSYMA system, 96, 571
  - domain knowledge, 251
  - knowledge engineering, 288
  - pattern matching, 718
- Maggs, P. B., 457
- Magnetic dipoles model, early vision, 1140
- Maida, A. S., 303–310
- Mail delivery, actor formalisms, 2
- Mailing lists, AI literature, 533
- Mainframes and computer chess programs, 160
- Mallery, J. C., 362–374
- Mal-rules, education applications of AI, 270
- Mammalian visual system, 493
- Management systems for medical advice systems, 588–589
- Mandala language, 557
- Manhattan Distance heuristic, 378, 997
- Manipulators, 571–575
  - biological systems, 924

- industrial robots, 945
- limits of, 496
- mobile robots, destination activities, 961
- natural-language user interfaces, 386
- position solution, 573–575
  - coordinate transforms, 574
  - trajectory planning, 574–575
  - vector loop approach, 573
- prostheses, 797, 799
- robots, 801–802
- robot configuration, 572–573
  - closed-loop two-input planar device, 573
  - cylindrical 3-D robot, 573
  - degrees of freedom, 572–573
  - joint-to-world coordinate transformation, 925
  - pin joint, 572
  - slider-type joint, 573
  - spherical configuration, 572
  - 3-D space, 572–573
  - two-link device, 572
- static-master-slave, teleoperators, 1100
- Manufacturing:**
  - continuous-flow manufacturing, 37
  - automation of operations:
    - make, 36
    - moving, 36
    - storing, 36
    - test, 36
  - continuous-process, 37
  - design for ease, 38
  - discrete-part, 37
  - hierarchical composition, 37
  - industrial robots, 954
  - integration, 37–38
  - just-in-time, 37
  - logistics, 37
  - robotics, 941
- Manufacturing Automation Protocol, robotics, 953
- Many-valued logic, 537
  - propositional logic and, 558–559
- MAPCAR system, LISP development, 520
- Map coloring, constraint satisfaction, 206, 209
- MAPLE system, 96
- Map matching, autonomous vehicle position location, 42
- Map transforms, vehicle route planning, 43
- Mapping:
  - path planning and obstacle avoidance, 715
  - route planning for mobile robots, 960
  - tree-to-tree deep structure, 231
- MAPSEE system, 399, 401
- Marcus parser, 360–361
  - language learnability models, 445
- LPARSFAL, 479
- Marcus, Ruth Barcan, 618
- MARGIE system:
  - conceptual dependency, 198
  - expectation-driven parsing, 696–697
  - machine translation, 570
  - POLITICS and, 757
  - script applications, 983–984
  - story analysis, 1092
  - text generation, 144–145
- Marine Corps, AI programs, 608
- Marine Integrated Fire and Air Support System (MIFASS), 608
- Marker passing and knowledge representation, 886–887
- MARK-GRAF program, 1120
- Markov process:
  - rule-based systems, 970
  - speech recognition, 1068–1069
  - texture analysis, 1105, 1107
- Marr-Poggio stereo model, 1087–1088
- Marr, David, epistemology, 285
- Marsland, T. A., 159–169
- Martins, J., 58–62
- Mass spectroscopy and chemical instrumentation, 95
- Masterman, M.:
  - machine translation, 569
  - semantic memory, 593
- MASTERSCOPE system, 523–524
  - INTERLISP and, 794–795
- Matched filtering, Hough transform, 383
- Matching:
  - aerial images, 577
  - artificial intelligence, 12
  - correlation and stereo vision, 1089
  - early vision, 1132–1133
    - segmentation, 1142
  - feature acquisition, 1086
  - frame theory memory organization, 305–306
  - image understanding, 392
  - information retrieval, 419
  - knowledge representation, 886
  - linear features, 577
  - pattern recognition, 728
  - stereo vision, 1086–1089
    - distance determination, 1088–1089
  - story analysis, script application, 1093
- Matching template, 576–577
  - applications, 577
  - basic mathematics, 576
  - chamfer technique, 577
  - coarse-to-fine, 577
  - difference of Gaussians (DOG) operator, 577
  - extensions, 576–577
    - robustness, 576–577
  - speed, 577
- Material conditional rule in predicate logic, 541
- Material requirements planning (MRP), 37
- Mathematics:
  - edge detection, 264
  - knowledge representation, 882
  - logic, 789, 822
  - LOGO, 269
  - programming assistants, 786
  - social issues of AI, 1050
  - teaching methods, 563
- Mathews, M., 638
- Mating and theorem proving, 1120
- Matrices:
  - saddle point in minimax procedure, 616–617
  - theorem proving, 1120
  - 2×2 payoff, 615–616
- Maxima selection, edge detection, 1137
- MBR, *see* Multiple belief reasoner
- MDX system, 585
- Meaning:
  - conceptual dependency, 197
  - congruence, hermeneutics, 373
  - deep structure and, 231
  - episodic memory, 278
  - hermeneutics, 363
  - propositional logic, 558
- Meaning representation in case grammar, 333, 338
- Means-ends analysis, 114, 476, 578–584
  - ABSTRIPS, 582
  - FDS program, 581
  - General Problem Solver (GPS), 323, 578–581
    - reduction schema, 775
  - good difference information, 583
  - hermeneutics, 369
  - MPS system, 582–583
  - pattern matching, 719
  - STRIPS program, 581–582
  - triangularity, 583
- Mechanization of Thought Processes*, 3
- Mechanized assistant, automatic program-
- MEDIATOR system, 592
- Medical advice systems, 584–590
  - belief revision, 61
  - case grammar, 333
  - causal reasoning, 827
  - clinical use, 590
  - decision theory, 229–230
  - diagnosis systems, 584, 587–588
    - educational applications of AI, 271
    - object-oriented programming, 453–454
  - example systems, 587–590
  - explanation, 586
  - knowledge acquisition, 586
  - management systems, 585, 588–589
  - pattern recognition, 727
  - reasoning, 823
    - inference applications, 824
  - research themes, 586–587
  - search methods, 994
  - social issues of AI, 1051
  - temporal reasoning, 586–587, 870
  - theoretical basis, 584–586
    - control, 585–586
    - evaluation function, 586
    - inexact inference, 586
  - knowledge representation, 584–585
  - protocol analysis, 584
  - validation, 587
- Meinong, Alexius, 282–283
- Meinongian logic, 541
- Meldman, J. A., 458
- Melkanoff, M. A., 153
- MEMBER program and MacLisp, 523
- Memory:
  - architectures, 16
  - associative, 16–18
  - cognitive modeling, 111, 113
  - cognitive psychology, 117–118
    - short- and long-term stores, 117
  - echoic, 276–277
  - episodic, 275–280
    - machine learning, 476–477
  - fetch reduction, 177
  - frequently used words in high speed stack, 177
  - game playing and, 317
  - iconic, 276–277
  - limitation in word-expert parsing, 705



- long-term:
  - cognitive modeling, 111
  - intelligence and, 435
- management, 176
- mental imagery, 117
- NON-VON computers, 678
- primary, 276–277
- quantitative estimates of, 489–490
- retrieval, 592
- script applications, 984
- scripts, 991
- semantic, 593–594
  - cognitive modeling, 113
  - direct extensions, 593
  - Quillian's models, 593
- semantic networks, 1012
- short-term, cognitive modeling, 111
- span, 277
- tables, computer chess methods, 167–168
- trace, 117, 278
- transformational grammar, 354
- Memory organization packets (MOPs), 591–592
  - E-MOPS, 992
  - emotion modeling, 274
  - frame theory, 305–306
  - integrated partial parser (IPP), 699
  - law applications, 461
  - scenes, 591–592
  - scripts, 591, 985
    - BORIS system, 989
    - psychological validity, 991
  - semantic networks, 1022
  - S-MOPS, 992
  - story analysis, 1097–1098
- Memory representation:
  - emotion modeling, 274
  - story analysis, 1091–1093
- MENO-II programming tutor, 155
  - human-computer interaction, 385
- Mental abilities, intelligence and, 433
- Mental informational transfer (MTRANS)
  - computational linguistics, 136
- Mentalistic psychology, 738
- Mental mechanisms (human) and cognitive modeling, 111
- Menu-based natural language (NLMenu),
  - 595–597
  - interfaces, 595–597
  - performance, 597
- Menu-based user interfaces, 386
- Merchant, M. E., 171–173
- Merging, color vision, 129
- Merleau-Ponty:
  - context and holism, 741
  - phenomenological hermeneutics, 367–369
  - phenomenology and, 731–732
- MERLIN system, 598
- Message passing:
  - actor formalisms, 2
  - control structures, 213
  - object-oriented programming, 454
- Message-Passing Semantics Group, 2
- Message refinement:
  - fluency, 648
  - input, 647–648
  - natural-language generation, 647–649
  - output, 648
  - specification, 648
- META-DENDRAL system, 94
  - concept learning, 190–191
  - inductive inference, 416
- Metagrammar, transformational grammar, 355
- Meta-knowledge, 598
  - connecting theories, 601
  - explanation and, 600
  - image understanding, 393
  - knowledge and, 600–601
  - motivations, 599–600
  - planning operation, 750
  - self-reference, 1007
  - systems with, 601–603
- Metalanguage, dialogue phenomena, 673
- Metamorphosis grammar (MG), 339
- Metaphysics, philosophy and AI, 736
- Meta-reasoning:
  - connecting theories, 601
  - medical advice systems, 587
  - motivations, 599–600
  - planning, 599–600, 750
  - vs. problem level, 823
  - reasoning and, 601
- Meta-rules, 598
  - cognitive phenomenon, 599
  - deduction and planning, 601
  - maintenance and acquisition of knowledge, 559–600
  - planning, 750
    - action selection, 753–754
  - PRESS system, 549
  - rule-based systems, 968
  - TEIRESIAS system, 602
- Metascience, 369–370
- Methodological solipsism, 732
- Methods procedures, object-oriented programming, 452
- Metropolis algorithm, constraint satisfaction, 210
- M-expressions, LISP development, 520–521
- Michalski, R. S., 103–110
- Microcomputers, social issues of AI, 1055–1056
- Microelectronics and Computer Corporation, 18
- Micro ELI, expectation-driven parsing, 699
- MICRO-PLANNER, 603–604
  - constraint satisfaction, 207
  - PLANNER and, 748
- Military applications of AI, 604–613
  - automated natural language understanding, 612–613
  - automatic test equipment (ATE), 610
  - autonomous land vehicle, 604
  - Built-in test (BIT), 610–611
  - combat resource allocation, 608–609
  - equipment maintenance and troubleshooting, 609–611
  - interactive maintenance aids, 611
  - mission planning, 609
  - naval battle management, 605–606
  - political control issues, 1056–1057
  - sensor information for situation description, 606–609
  - training, 611–612
- Military robots, 954
  - mobile, 961
- Miller, D., 299
- MIND system, propositional networks, 1015–1016
- Minimal window search, computer chess methods, 162
- Minimax procedure, 614–617
  - checkers programs, 89
  - computer chess methods, 161
  - demon inheritance, 429
  - game playing:
    - search reduction, 315–316
    - strategies, 313–315
  - game trees, 319
  - heuristics, computerized chess, 377
  - horizon effect, 380–381
  - specialization, 615–617
    - three-step game, 616–617
  - strategies and payoffs, 615–616
  - two-person games, 378–379
- Minimum spanning tree, dot-pattern analysis, 255
- Minsky, Marvin, 1052
  - connectionism, 204
  - emotion modeling, 274
  - frame theory, 302–310
  - on intelligence, 790
  - neat/scruffy debate, 537
  - nonmonotonic reasoning, 849
- Misplaced concreteness, 371–372
- Misrecognition in frame theory, 304
- MITalk text-to-speech system, 1072
  - evaluation, 1075
  - letter-to-sound rules and lexical stress, 1073–1074
  - morphology, 1073
  - prosodic framework, 1075
  - stress modification and phonological adjustments, 1074
  - synthesis, 1074
- MIT LISCOM, 521
- MIT LISP Machine, human-computer interaction, 385
- MITRE Corporation, 606–609
- Mitre parser, 142
- MIT ZETALISP, 529
- Mixed initiative dialogue:
  - computer-aided instruction, 154
  - natural-language automatic programming, 30
- MMU system, associative memory, 17
- Mobile robots, 924, 957–961
  - applications, 961
    - industrial, 961
    - nuclear, 961
    - undersea, 961
  - beacon systems, 958
  - computing architectures, 960–961
  - destination activities, 961
  - docking, 961
  - landmark systems, 958–959
  - legged locomotion, 957–958
  - passive suspension, 957
  - position finding, 958
  - preprogrammed path guidance, 959
  - proximity sensing, 804
  - route finding, 959
  - route planning, 959–960
  - semiautonomous suspension, 957
- Mobility aids, 797–799

- Modal logic, 537, 617–619  
 dynamic, 619  
 first-order, 618–619  
 formalisms, 617–618  
   axiomatic systems, 618  
 Kripke models, 618  
 predicate logic and, 538  
 propositional attitudes, 837–838  
 temporal logic, 618
- Mode-based parameter estimation, 383
- Model-based vision, shape analysis, 1046–1047
- Model recognition, database organization, 634–636
- Model-theoretic semantics (MTS), 760, 1026
- Model theory, 547
- Modeling:  
   cognitive, 111–114  
   computer-aided design, 151–152  
   earth's surface, 1106  
   electronic, dimensionality, 36  
   engineering design, 36  
   human-computer interaction, 385  
   object, industrial automation, 39  
   organization, word-expert parsing, 705  
   qualitative physics, 810  
   robotics programming, 940  
   texture analysis, 1102–1111  
     autoregressive models, 1103, 1107  
     circulant matrices, 1107–1108  
     conditional probability, 1106–1107  
     Gaussian noise, 1105  
     global models, 1104  
     joint probability approach, 1108  
     local models, 1106  
     Longuet-Higgins model, 1105–1106  
     mixed model, 1103–1104  
     moving-average model, 1103  
     one-dimensional time-series models, 1103–1104  
     random-field models, 1104  
     region-based, 1109–1110  
       coverage models, 1109–1110  
       mosaic, 1109–1110  
     syntactic models, 1108  
     unknown parameters, 1107
- Modified constraint, backtracking, 46
- Modifier structure grammars (MSGs), 339, 342
- Modularity, control structures and, 212–213
- Modula 2, recursion, 875
- Modus ponens*:  
   belief systems, 67  
   binary resolution, 894–895  
   knowledge representation, 884–886  
   nonmonotonic reasoning, 851  
   reasoning, 823  
     inference application, 824  
     uncertainty, 854
- Moire patterns, dot-pattern analysis, 255
- Molecular propositions, 559–560
- MOLGEN system:  
   constraint satisfaction, 210  
   description, 602–603  
   planning applications, 756  
   macro action, 754
- Moment-by-moment optimization, computer-integrated manufacturing, 171
- Monitor system, question answering, 818
- Monochrome imaging, color vision, 124–126
- Monocular vision, SHAKEY robots, 1039
- Monotonic logic:  
   artificial intelligence, 13  
   default reasoning, 844
- Montague grammar:  
   commonsense reasoning, 833  
   discourse understanding, 239  
   generalized phrase structure grammar, 343  
   knowledge-base semantics, 1028  
   lambda calculus, 442  
   machine translation, 565  
   semantic networks, 1019  
   semantics, 1026
- Moon, David, 523
- Moore, G. E., 283
- Moore, J., 302, 1126–1130
- MOPTRANS, 592  
   expectation-driven parsing, 700–701
- MORAN program, semantic networks, 1022
- Moravec's matching scheme, distance determination, 1088–1089
- Morawski, Paul, 604–613
- Morgado, E., 598–603
- Morphology, 619–620  
   analysis, 620  
   linguistic competence, 503–504  
   speech recognition, 1068  
   speech synthesis, 1072–1073  
   speech understanding, 1077
- Moser, M. G., 339
- Most general unifier (MGU), theorem proving, 1119
- Motion analysis, 620–630  
   accuracy of algorithms, 629–630  
   early vision, 1143–1144  
     rigidity assumption, 1144–1145  
   edge detection, 266  
   flow fields, 626–627  
   future research, 630  
   high-level motion understanding, 630  
   image intensity matching, 627–628  
   limits of artificial intelligence, 496–497  
   modeling and prediction, 630  
   multiple objects, 630  
   nonrigid objects, 630  
   optical flow motion, 624–625  
   orthographic projections, 624  
   planar curve correspondence, 624  
   robot motion-planning, 866  
   small-motion case analysis, 624  
   3-D feature correspondences, 628–629  
   three-view case, 623–624  
   two-view motion with feature correspondences, 621–624  
     point correspondence, 621–622, 621–624
- Motion correspondence, optical flow, 684
- Motivation, ICAI and, 158
- Motor control:  
   connectionism, 200, 204  
   limits of artificial intelligence, 496–497
- Move generation, computer chess, 165–166
- MOVER program, path planning and obstacle avoidance, 709–710
- MOW knowledge source, 75
- MPP machine, symbolic computation, 216
- MPS system, means-ends analysis, 582–583
- MRS system, self-reference, 1007–1008
- MS.MALAPROP, 632
- Mueller-Lyer illusion, vision, 1132
- MULTIBUS, LISP machine, 529
- Multipass algorithms, coroutines, 220
- Multiple belief reasoner (MBR)  
   belief revision, 58, 60  
   changing belief sets, 60–61  
   faulty assumptions, 61
- Multiple instruction stream, multiple data stream (MIMD), 678
- Multiple representations, object-oriented language, 454
- Multiple senses (polysemy), 367–368
- Multiprocessor chess programs, 169
- Multisensor integration, 632–637  
   biological systems, 924  
   database access, 636  
   database model organization, 634–636  
     boundary curves, 636  
     cavities, 636  
     gross shape, 634  
     holes, 636  
     surface description, 634, 636  
   depth-map reconstruction, 1154  
   discrimination features, 634  
   industrial robots, 946  
   sensor environment, 633–635  
   sensory data convergence, 637  
   strategies, 636–637
- MUMBLE system:  
   natural-language generation, 647–649  
   text generation, 145
- Munsell color orders system, 126–127
- Music:  
   AI in, 223, 638–641  
   constrained selection, 639–641  
   contextual pattern selection, 638–639  
   education, 183
- Musical composition and creativity, 223
- MUSIC11 program, 638
- MUSICV program, 638
- Mutual belief theory, 69
- MYCIN system, 641  
   certainty factor, 14  
   confirmation theory, 856–857  
   default reasoning, 846  
   domain knowledge, 252  
   education applications, 268  
   explanation, 299  
   functions and design, 589–590  
   history of, 10, 587–588  
   intelligent computer-aided instruction (ICAI), 155  
   knowledge acquisition, 586  
   knowledge engineering, 288, 296  
   knowledge representation, 888  
   law applications, 457–458  
   rule-based systems, 971  
   theoretical issues, 585–586  
   Turing test, 1129
- Mylopoulos, J., 882–889
- MYRTILLE-II, speech understanding, 1081
- Nagel, R. N., 944–954
- Naive physics, 497, 746–747  
   causal reasoning, 834  
   qualitative physics and, 807, 809

- reasoning and, 869
- temporal reasoning, 871–872
- Naive physiology, 1136–1137
  - early vision, 1144
- NAMER system, computational linguistics, 136
- Nano instructions, connection machines, 179
- NANOKLAUS system, 186
  - semantic networks, 1022
- NAOS system, 402
- Narrative analysis, CASREP system, 612–613
- Narrative inference emotion modeling, 274
- Narratives, dynamic discourse model, 237
- National Bureau of Standards, industrial robots, 949
- Natural-language generation, 642–654
  - character, 643–644
  - common approaches, 647–650
  - components and terminology, 644–645
  - explanation, 299–300
  - grammar treatments, 650–653
    - functional unification grammar (FUG), 650–651
    - surface structure as intermediate representation level, 651–652
    - systemic grammar and ATNs, 652–653
  - historical perspectives, 646–647
  - lexical choice, 649
  - message refinement, 647–649
  - phrasal lexicons, 649–650
  - planning, 653–654
    - psycholinguistic theory, 653–654
  - state of the art, 645–646
- Natural-language interfaces, 386, 655–659
  - ATNs, 324
  - conjunction and disjunction, 659
  - ellipses, 658
  - end-user control of interpretation, 658
  - English as appropriate language, 656
  - human-computer interaction, 383–384
  - ICAI systems, 154
  - inference, 658
  - negation, 658
  - procedural semantics, 1030
  - pronoun use, 658
  - prospective users, 657–659
    - coverage and habitability, 657–658
  - quantification, 658
  - question answering, 814
  - semantic grammar, 351
  - speech synthesis, 1070
  - state of the art, 656–657
  - telegraphic input, 659
  - time and tense, 658–659
  - ungrammatical input, 659
- Natural-language processing (NLP), 660
  - as AI technique, 14
  - case grammar, 333
  - circumscription and, 102
  - cognitive modeling, 113–114
  - commonsense reasoning, 833
  - computational linguistics, 133
  - deep structure, 231
  - hermeneutics, 362
  - machine translation, 564
  - nature of, 661
  - question answering, 814
  - story analysis, 1091
- Natural language systems:
  - computational linguistics, 134–135
  - medical advice systems, 585–586
  - RELATUS, 373
  - speech acts, 1062
- Natural-language understanding, 660–675
  - anthropomorphic robots, 945
  - artificial intelligence, limits of, 494–496
  - augmented transition network (ATN)
    - grammar, 324
  - automatic program synthesis, 19, 29–32
    - example dialogue, 30
    - functional modules: acquisition, 30–31
    - functional modules: coding, 31–32
    - historical remarks, 32
    - implementation, 32
    - system design, 29
  - belief revision, 61
  - case frame instantiation, 668–671
  - CASREP, 612
  - chemistry, 93–94
  - cognitive modeling, 111
  - commercialization, 141
  - connectionism, 200, 204
  - database interface, 137–138
  - definite-clause grammar, 339–340, 342
  - dialogue phenomena, 672–675
    - anaphora, 672
    - case frame ellipsis, 673–674
    - definite noun phrase, 672–673
    - ellipses, 673
    - extragrammatical utterances, 673
    - indirect speech acts, 673
    - metalinguistic utterances, 673
  - discourse understanding, 233
  - domain knowledge, 251
  - epistemology, 285
  - expectation-driven parsing, 696
  - explanation, 299
  - generation, 643
  - goal determination inference, 675
  - grammar, 411
  - hermeneutics, 371
  - information retrieval, 420–421
  - interface, *see* Natural language interface
  - knowledge representation, 882
  - law applications, 461
  - limits of, 491, 493
  - logic programming, 546
  - menu-based, 595–597
    - interfaces, 594–597
    - performance, 597
  - meta-knowledge, 598
  - military applications of AI, 605–606
  - MS.MALAPROP, 632
  - pattern matching, 716, 718
  - phenomenology, 741
  - predicate logic, 538
  - problem solving, 768
  - problems with conventional, 594–595
  - procedural semantics, 1030
  - question answering, 814
  - reasoning, 848–849
  - robust parsing, 671–672
  - scripts, 985–990
  - semantic grammar, 351, 667
  - semantics, 1026
  - SHRDLU, 1048
  - SIR system, 1048
- social issues of AI, 675, 1052
- speech acts, plan-based models, 1063
- syntactically-driven parsing, 664–665
- techniques, 663–664
- theorem proving, 1117
- word-expert parsing, 701, 707
- Natural science and hermeneutics, 370
- Nau, D., 316
  - game playing techniques, 316–317
- Naval battle management and artificial intelligence, 605–606
- Naval Research Laboratory, 609
- Navigation, industrial robots, 953
- NCOMPLR system, 521
  - MacLisp, 523
- Neat/scruffy debate, 537
- Necessity and possibility theory, 858
- Needle maps, 1153
- Needs analysis and computer-integrated manufacturing, 171–172
- Negation:
  - as failure, inheritance hierarchy, 426
  - natural-language interface, 658
- Negative information and inference, 419
- Negative-horizon effect, 380–381
- Neighborhoods, dot-pattern analysis, 255
- Nemes, 159
- Neomata, 732
- NEOMYCIN system, 586
  - explanation, 299
- NETL system:
  - connectionism, 204
  - frame language, 306, 309
  - inheritance hierarchies, 422, 424, 426–427
    - splits and partitioning, 428
  - knowledge representation, 887
  - implementation, 1022–1023
  - VC relation, 422
- Networks:
  - construction, 203–204
  - recursive transition (RTN), 324–325
  - robots, 953–954
- Neurology:
  - brain activity, 489
  - cognitive science, 122
  - connectionism, 200–201
  - language competence-performance dichotomy, 506
  - modeling and early vision, 1139
  - social issues of AI, 1050
- Neuropsychology, intelligence and, 437
- Newell, A., 9, 132, 302, 1052
  - game playing, 316
  - GPS, 323
- Newmeyer, F. J., 503–508
- Newsletters, artificial intelligence, 533
- NEWTON system for physical reasoning, 869
- Newton's law:
  - qualitative physics, 812
  - robot-control systems, Lagrangian formulation, 909–911
  - single-degree-of-freedom robot arm, 903
- Newton-Euler formation, robotics manipulators, 926
- NEXT operator, beam search, 56
- Nexus and knowledge representation, 889

- Neyman-Pearson scheme, pattern recognition, 722
- NIGEL system, 653
- text generation, 143, 145
- NIL system:
- development, 525
  - history, 526
- Nilsson, N. J.
- frame theory, 302
  - heuristic belief system, 66–68
- Nitza, D., 923–941
- NLC system and office automation, 140
- NLPQ system:
- automatic program synthesis, 32
  - computational linguistics, 140–141
  - office automation, 140
- No-function-in-structure principle in causal reasoning, 829
- NOAH system, 602, 677, 750
- distributed problem solving, 249
  - hierarchical planning, 754
  - historical context, 757
  - planning cycle, 601
  - problem solving, relaxed reduction, 775
- Nodes:
- checkers programs, 88
  - planning, 750–751
- Noisy AND gates, Bayesian decision methods, 54
- Noisy data, concept learning, 473
- Noisy OR gates, Bayesian decision methods, 54
- Nondeterminism:
- ATNs, 327, 329
  - computer-integrated manufacturing, 173
  - control structure, 213
  - left-corner parsing, 690
  - parsing, 142
  - recursive-descent parsing, 689–690
  - temporal reasoning, 874
- Nonexistent object logic, 537
- NONLIN planner system, 750
- applications, 756
  - hierarchical planning, 754
  - historical context, 757
- Nonlinear equations, motion analysis, 622
- Nonlinear operators, feature extraction, 301
- Nonlinear systems theory, robot-arm control, 916–919
- Nonmonotonic logic, 849–852
- applications and related work, 852
  - belief revision and, 58
  - circumscription, 100
  - commonsense reasoning, 834
  - default reasoning, 842–843
  - formalisms, 850
  - inheritance hierarchies, 422–423
  - knowledge representation, 889
  - McDermott and Doyle, 850–851
  - Moor theory, 851
  - Reiter's default logic, 851
  - system, 537
  - temporal reasoning, 872
  - underlying reasoning, 851–852
- Nonparametric shapes, Hough transform, 382
- NON-VON computers, 678–679
- applications and performance evaluation, 678–679
  - computer architecture, 216
- Normal-form grammars, 689
- Normalized texture property map, shape analysis, 1046
- Norman, D., 204, 302
- Norvig, P., 304
- NOTICE-PATTERN knowledge system, 77
- n*-Queens problem, constraint satisfaction, put under Queens 209
- N* tables, minimax procedures, 615
- NTSC color TV standards, 126
- n*-tuple Search, route search, 43
- Nuclear magnetic resonance, 288
- chemical structure elucidation, 94–95
- Nuclear weapons and social issues of AI, 1056
- NUDGE system, 312
- Numbers:
- LISP, run time typing, 518
  - LISP 1.5, 521
- NUMTAX system and clustering, 108
- Nutter, J. T., 280–285, 840–846
- O'Rourke, P., 304
- Oak Ridge and Federal Translation Division, 566
- Object-level rules, 549
- Object-oriented language
- domain knowledge, 253
  - frame theory, 309–310
  - ICAI, 157
- Objected-oriented programming, 452–456
- connection machines, 200
  - control structures, 212–213
  - coroutining, 219, 222–223
  - diversity, 455–456
  - education applications of AI, 269
  - environment, 791–792
  - inheritance, 422, 429–430
  - intelligent programming, 452–453
  - knowledge sharing, 454–455
  - lambda calculus, 442
  - LISP Machine flavors system, 526
  - logic programming, 553, 556–557
  - medical advice systems, 453–454, 585
  - message-passing behavior, 454
  - origins, 455
  - parallelism, 455
  - SMALLTALK language, 1049
  - ZETALISP, 529
- Object-recognition systems
- multisensor integration, 632
  - model-based, 633–635
- Objectivity and hermeneutics, 366, 371–372
- Objects:
- database, 891–892
  - modeling:
    - industrial automation, 39
    - intelligence and, 432
  - object-oriented language, 452
  - properties, 195
  - measurement, 929–930
  - recognition
    - connectionism, 204
    - multisensor integration, 632
- Obstacle avoidance, 708–715
- autonomous vehicles, 42–43
  - configuration space, 710–711
  - convex decomposition, 712
  - derivation-formation spectrum, 771
  - free-space approaches, 711–712
  - potential fields, 713
  - problem definition, 708
  - route finding, mobile robots, 959
  - static knowledge and dynamic knowledge, 708–710
  - taxonomy of techniques, 709
- Obstacle detection, autonomous vehicles, 40
- OCCAM program, script acquisition, 992
- Occam's Razor, default reasoning, 843
- Occupancy rays, volume description, 864
- OCEAN system, 972
- Oct-trees, 835
- Odetics vehicle, 928
- Odex robot, 949
- Office automation, 680–683
- education applications of AI, 681
  - communications, 682
  - information domains, 682
  - machines, 681–682
  - organization, 682–683
  - overhead domains, 680–681
  - programming assistants, 786
  - routine and nonroutine activity, 680
  - subject domains, 680
- Office of Naval Research, 604
- Ohlander, R., region splitting, 878
- Omega system, inheritance hierarchies, 422
- ONCOCIN system, 587
- functions and design, 589–590
  - office automation, 681–682
- One-variable pattern language, 414
- Online help, office automation, 681
- On-line help system, question answering, 820
- Ontology:
- hermeneutics, 374
  - predicate logic, 540
- OpEd system script, 981, 990
- Open Systems Interconnect (OSI), robotics, 953
- Operationalism, Turing test, 1127
- OPM system, blackboard architecture, 73, 77
- OPPLAN-CONSULTANT system, 609
- OPS-4, 684
- OPS-5, 95, 684
- knowledge representation, 888
  - LOOPS program, 564
  - path planning and obstacle avoidance, 715
  - pattern matching, 719
  - performance evaluation, 678
  - reasoning, 823
- OPS system:
- domain knowledge, 252
  - knowledge engineering, 297
  - rule-based systems, 971
- Optacon reading machine, 800–801
- Optical character recognition (OCR), shape analysis, 1044
- Optical flow, 684–686
- connectionism, 204
  - edge detection, 266

- image velocities, 686
- measurement, 684–685
  - aperture, 684–685
- motional analysis, 624–625
- obstacle detection, 40
- proximity sensing, 805
- robotics sensing, 935
- shape analysis, 1040
- stereo vision, 1083
- uses, 686
- Optical parameters, mobile robots, 958
- Optical processing, template matching, 577
- Optimal solution graph, problem reduction, 764–765
- Optimization:
  - backtracking, 46
  - constraint satisfaction, 209–210
  - depth map reconstruction, 1157
  - knowledge systems, 290
  - problem solving, 777
  - programming assistants, 785
  - programming contraction, 793–794
- Ordering and planning, 751–753
  - goals, 755
- Organization:
  - episodic memory, 278
  - office automation, 682–683
  - scale and, 974
  - scale-space imaging, 977
  - structuring, distributed problem solving, 248–249
- Orientation, curve and flow recovery, 1146–1147
- Orthogonal mesh, NON-VON computers, 678
- Orthogonal slice-tracing, multisensor integration, 634, 636
- Orthographic projections, 624
  - motion analysis, 624–625
- Ostrich system, 169
- Othello (game), heuristics, 378
- Outcomes and decision theory, 229
- Outlier detection, multisensor integration, 636–637
- Output presentation, natural language interface, 659
- Overton, K., 1031–1036
- OWL system, 458
  - knowledge representation, 889
  - semantic networks, 1019
- PAIRS system, 95
- PAM (Plan Applier Mechanism), 687
  - discourse understanding, 240
  - plan recognition, 242
- PHRAN and PHRED, 746
- story analysis, 1094–1095
- world knowledge acquisition, 1096
- PANDEMONIUM, 232, 687
  - rule-based systems, 971
- Paper work flow and office automation, 681
- Papert, S., 181–185
  - connectionism, 204
  - frame theory, 302
  - object-oriented language, 455
  - perceptrons, 730
- Parabelle program, 169
- Paradise program, computer chess, 168
- Parallelism:
  - automatic programming assistance, 33–34
  - beam search, 57
  - blackboard systems, 79
  - Boltzmann machine, 80–81
  - cognitive psychology, 116–117
    - early and late models, 117
  - computer architectures, 217
  - connection machine, 177–178, 199–200
  - coroutines, 219
  - DADO, 227–228
    - language, 228
  - demons, 232
  - depth map reconstruction, 1156
  - early vision, 1139
  - logic programming, 555–556
  - object-oriented programming, 455
  - parsing, 142
  - semantic memory, 594
  - speech understanding, 1080
  - von Neumann architecture, 174
- Parameterized curves, Hough transform, 382
- Parametric representation, speech synthesis, 1071
- Paramodulation, 418, 498
  - theorem proving, 1117–1118
- ParaPhoenix program, 169
- Parikh, R., 617–619
- Paris Exhibition of 1914, 159
- PARLOG, 556–557
- PARRY system, 687
  - belief system, 70
  - emotion modeling, 273–274
  - natural-language understanding, 663–664
- PARSE program, processing, bottom-up and top-down, 780
- Parse trees:
  - inductive inference, 414
  - syntactic analysis, 664–665
  - transformational grammar, 666
- PARSIFAL system:
  - ATN grammar, 329–330
  - deep structure, 231
  - transformational grammar, 360–361
- Parsimony, natural-language generation, 650
- Parsing, 687–695
  - augmented transition network grammar, 330, 693–694
  - automatic program synthesis, 29
  - case frames, 670–671
  - case grammar systems, 335–336
  - chart, 691–692
  - CKY, 691–692
  - cognitive science, 121
  - complex feature-based grammar, 692
  - computational linguistics, 141–143
    - direction of analysis, 142
  - conceptual dependency, 199
  - coroutining, 219
  - data-driven, word-expert parsing, 702
  - deep structure, 231
  - Earley algorithm, 327, 330
  - empirical speed comparisons, 694–695
  - expectation-driven, 696–701
    - processing, bottom-up and top-down, 784
  - semantic networks, 1021
- fuzzy, semantic grammar, 352
- inductive inference method, 412
- integrated in-depth, scripts, 988–989
- language learnability models, 445
- left-corner, 690–691
  - processing, bottom-up and top-down, 784
- LIFER system, 488
- limits of artificial intelligence, 495
- logic programming, 546
- music applications, 638
- natural-language, 663
  - ATNs, 652
  - reasoning, 823
- normal-form grammar, 689
- pattern matching, 663–664
- pattern recognition, 720, 725
- phrase-structure grammar, 688–689
- processing, bottom-up and top-down, 779, 781
  - morsel and target strategies, 780
- recursive descent, 689–690
  - processing, bottom-up and top-down, 780
- recursive transition networks (RTNs), 324
- robust:
  - construction-specific approach, 672
  - natural-language understanding, 671–672
- search and nondeterminism, 142
- semantic networks, 1021
- semantics and syntax, 1024–1025
- simple sentences, 479
- speech synthesis, 1074
- speech understanding, 1077
- strategies for competence and performance, 507
- syntax-directed, 664
  - semantic networks, 1021
- transformational grammar, 692–693
- word-based systems, 142
- word-expert, 701–707
  - semantic networks, 1021
- Partial differential equations (PDE), texture analysis, 1105
- Partial matching, multisensor recognition, 637
- Parzen window, pattern recognition, 723
- Pascal language:
  - GPSG parsers, 343
  - industrial robots, 950
  - limits of, 501
  - pattern matching, 716
  - recursion, 875
  - semantic networks, implementation, 1022
- Path graph nodes, map transforms, 43
- Path planning, 708–715
  - C-cells, 712
  - configuration space, 710–711
  - decision point generation, 710
  - derivation-formation spectrum, 771
  - exhaustive search, 710
  - following, 713–714
  - generalized cylinders, 712
  - Heuristic (A\*) search, 710
  - industrial robots, 952–953
  - problem definition, 708
  - route planning, mobile robots, 960

- static knowledge and dynamic knowledge, 708–710
- taxonomy of techniques, 709
- Path relaxation and obstacle avoidance, 713
- Patient-specific model (PSM), medical advice systems, 588
- Pattern matching, 716–719
  - algebraic expressions, 718
  - algorithms, 716
  - approximate matching, 717
  - computer systems, 174
  - DADO, 228
  - expert-system shells and AI languages, 719
  - ICAI, 719
  - machine translation, 568–569
  - music and AI, 638
  - natural-language analysis, 663–664
  - pattern-directed invocation, 718
    - knowledge representation, 884–885
    - inference, 886
  - pattern-directed retrieval, 717–718
  - problem solving and planning, 719
  - processing, bottom-up and top-down, 784
  - program transformation and synthesis, 719
  - rule-based systems, 968
  - script, 981–982
  - SNOBOL-4 language, 1049
  - speech recognition, 1066–1067
  - speech understanding, 1076
  - structures, 717
  - template matching, 717
  - unification, 717
  - variables use, 716–717
- Pattern Recognition*, 721
- Pattern recognition, 720–728
  - anthropomorphic robots, 945
  - applications, 727–728
    - earth and space sciences, 728
    - fingerprint and character recognition, 728
    - medical information processing, 727
    - radar and sonar, 728
    - speech processing, 727–728
  - chemistry, 93
    - fragment determination, 94
  - cluster analysis, 103
  - color vision, 128
  - decision theory, 721–723
  - feature analysis, 83
  - feature extraction, 300–301
  - game playing, 317
  - human-computer interaction, 384
  - inductive inference, 409
  - iterative development, 722
  - machine learning, 464–465
  - models and methodologies, 721–727
  - multistage classification, 723–724
  - music applications, 638
  - nonparametric techniques, 723
  - parameter estimation, 722–723
  - pattern matching, 716
  - perceptrons, 730
  - performance assessment and feature selection, 726–727
  - point estimation, 722–723
  - problem-reduction representations, 726
  - signal processing, 721
  - speech recognition, 1065
  - state-space models, 724
  - syntactic, 716, 724–725
  - texture analysis, 1102
  - unsupervised methods, 723
  - Viterbi algorithm, 1160
- Pauker, S., 229–230
- Paxton parser, 330
- PDP-1 and LISP development, 522
- PDP-6, 522
- PDP-10, 522
  - computer architecture, 215
  - control, 928
  - demise of, 524–525
  - MacLisp, 523
- PDS system, computer-integrated manufacturing, 173
- Pearl, J., 7–8, 82, 317, 319–321
- Pedagogical methods, cognitive science, 120–121
- Pellegrino, J., 436
- Perception:
  - cognitive modeling, 111–113
  - cognitive psychology, 116–118
  - constraint satisfaction, relaxation algorithms, 209–210
  - cycle of image understanding, 394–395
  - dot-pattern analysis, 254
  - epistemology, 283
  - frame theory, 303–304
  - functionalism and, 738
  - hermeneutics, 373
  - hermeneutics and, 364–365
  - multi-sensor integration, 633
  - path planning and obstacle avoidance, 715
  - scale-space filtering, 974
  - texture analysis, 1101
  - Viterbi algorithm, 1160
- Perceptron, 729–730
  - connectionism, 204
  - convergence procedure demons, 232
  - diameter-limited, 730
  - learning algorithm:
    - game playing, 317
    - pattern recognition, 723
  - practical limitations, 729–730
  - theoretical issues, 730
- Perceptual grouping, dot-pattern analysis, 253–254
- Perceptual interpretation, Boltzmann machine, 80–81
- Perfect-information games, 319–321
- Performance:
  - control structure, 214
  - creativity, 225
  - defined, 504–505
  - engineering, programming tools, 794
  - evaluation:
    - information retrieval, 421
    - NON-VON computers, 678–679
    - pattern recognition, 726–727
  - ICAI, 157–158
  - linguistic, 503
  - musical, AI and, 638
- Periera, F., 339
- Peripheral rollers, semiactive suspension, 957
- Perlis, D., 100–102, 849–852
- Perrault speech act theory, 68
- Persistences in temporal reasoning, 872
- Personal relations, 838
- Personal robots, 954
- Peterson, M. A., 459–460
- Petrick parser, 142, 691
  - transformational grammar, 360
- Petrick, S., 687–695
- Petri nets, actor formalisms, 2
- Phantom flight crews, military AI, 604–605
- Phase space, single-degree-of-freedom robot arm, 903
- Phase table method, checkers-playing programs, 92
- Phenomenology, 730–735
  - AI and human expertise, 731–735
    - advanced beginner, 734
    - competence, 734
    - expert level, 734–735
    - novice stage, 733
    - proficiency, 734
  - context and holism, 741
  - critique of, 740–742
  - eidetic, 373
  - episodic memory, 275–276
  - epistemology and, 283
  - hermeneutics, 364, 367–369
  - transcendental and existential, 731
- Phi-naught system, programming assistants, 787–788
- Philosophical Investigations*, 741
- Philosophy, 736–743
  - epistemology, 280–286
  - hermeneutics and, 364–365, 374
  - of mind, 736–737
  - ordinary language, 283, 366
  - phenomenology and, 731
  - semantics and, 1025
  - social issues of AI, 1050
  - speech acts, 1062
  - temporal reasoning, 873
  - Turing machines, 1125
  - Turing test, 1127
- PHILOSOPHY-OF-SCIENCE mailing list, 533
- PHLIQA system, database interface, 138
- Phonemes, 744–746
  - artificial intelligence, limits of, 493–494
  - distinctive features, 745–746
  - image understanding, 390
  - plural rule, 744–746
  - rhymes, 744
  - speech recognition, 1065, 1067
  - speech understanding, 1076
- Phonology:
  - morphological analysis, 620
  - speech understanding, 1079
- Photometric properties:
  - feature extraction, 301
  - shape from shading, 1043–1044
- PHRAN and PHRED programs, 746
  - semantic grammar, 350
- Phrasal lexicons:
  - natural-language generation, 649–650
  - recursive layers, 653
- Phrase structure grammars, 344–350
  - ATNs, 325
  - automata and, 689
  - EPISTLE system, 287
  - formal properties, 345–346



- generalized, 347–348
- linguistic theory, 444
- nonterminal vocabulary, 688
- parsing, 688–689
- revival, 347–348
- rules or productions, 688
- start symbol, 688
- terminal symbols, 346
- terminal vocabulary, 688
- transformation grammar, 346–347, 355
- tree-adjointing grammar (TAG), 348–350
- trees, 344
  - revival, 347–348
  - X-bar theory, 358–359
- Physical reasoning, 737–738, 835, 869
- Physical symbol system hypothesis, 9
- cognitive science, 123
- Physician's Desk Reference*, 453
- Physics:
  - commonsense reasoning, 836
  - multiple theory, 836
  - naive, 746–747
  - qualitative, 807–813
  - qualitative vs. quantitative, 807–808
- Piaget, Jean:
  - causal reasoning, 831
  - development of intelligence, 437
  - language acquisition, 444
  - perceptrons, 730
- Piano-movers problem, 710–711
- Picture aides, 195
- Picture arrangement, 432
- Picture completion tasks, 432
- Picture producers, conceptual dependency
  - rules, 195–196
- Pierce, T., 93–97
- Pilot's Associate system, 604–605
  - political issues, 1056
- PIM-R, computer architecture, 216
- Pinker, S., 445
- PIP system:
  - domain knowledge, 252
  - medical advice systems, 585
- Pipelining:
  - computer architecture, 215
  - computer systems, 174
  - coroutines, 220
- Pitrat, J., 315
- Pitts, W., 204
- Pixels:
  - robot vision, 1033–1034
  - volumetric description, 863–864
- PLACE system, industrial robots, 947
- PLA system, associative memory, 17
- Plan recognition:
  - speech acts, 1063
  - story analysis, 1094
- Planar curve correspondence, 624
- Planar patches:
  - linear algorithm, 628
  - motion analysis, 622–623, 626
- PLANES system, 748
  - case frame ellipsis resolution, 673
  - database interface, 138
  - semantic grammar, 350
  - semantic memory, 593
- Planetary exploration:
  - mobile robots, 961
  - semiactive suspension, 957
- Planner-control coordination, autonomous
  - vehicle control, 41
- PLANNER system, 546, 748
  - antecedent theorems, 887
  - belief revision, 58
  - CONNIVER, 205
  - consequent theorems, 887
  - control structure, 213
  - erasing theorems, 887
  - knowledge representation, 883, 887–888
  - MICRO-PLANNER, 603–604
  - pattern matching, 718
  - programming environment, 792
  - SHRDLU and, 1048
- Planning, 748–757
  - action representation, 748–750
  - action selection, 753–754
  - advanced topics and research questions, 756–757
  - backtracking, 47
  - belief revision replanning, 61
  - blackboard architecture, 77
  - blocks world, 748
  - commonsense reasoning, 833
  - computer-managed, limits of, 498
  - constraints, 754
  - deep vs. shallow knowledge, 755
  - definite-clause grammar, 339
  - derived assertions, 749
  - distributed, 246
  - epistemology, 284
  - explanation and, 600
  - game-playing programs, checkers, 88
  - generalized plans, 478–479
  - goal hierarchies and ordering, 755–756
  - heuristic critics, 757
  - heuristic evaluation, 377
  - hierarchical, 754–755
  - image understanding, 400
  - industrial robots, 952–953
  - inhibiting backward chaining, 754
  - intelligent agents, 837
  - law applications, 460
  - means-ends analysis, 582
  - meta-knowledge, 598
  - natural-language generation, 643–644, 653–654
  - natural-language interfaces, 655
  - operation, 750–753
  - opportunistic, 755
  - path planning and obstacle avoidance, 708
  - pattern matching, 719
  - plan and goal structure, 837
  - practical applications, 756
  - problem solving, 767
    - relaxed reduction, 775
  - procedural subplanners, 755
  - process, industrial automation, 38
  - question answering, 816–818
  - spatial knowledge and inference, 835
  - speech acts, 1063–1064
  - temporal reasoning, 756, 871
- Plato (philosopher), 63
  - epistemology, 281
- PLATO system, 267
  - intelligent computer-aided instruction, 155
- Plausible reasoning, 854–862
  - belief theory, 857–858
- confirmation theory, 856–857
- uncertainty and, 854–855
- PL/I semantic networks, implementation, 1022
- Plot units, hermeneutics, 371
- Pohl, I., 1000
- Point correspondence, motion analysis, 621–622
- Pointer information, LISP, 517
- Pointing device:
  - programming tools, 792–793
  - visual communication aids, 803
- Poisson-Brown surface model, 1106
- Poker and mathematical formulation, 312
- POL system and database interface, 138
- Polansky, L., 638
- Polanyi, L., 233–244
  - discourse understanding, 233–244
- Pole placement technique, single-degree-of-freedom robot arm, 906
- Polish notation, propositional logic, 560
- POLITICS system, 757
  - story analysis, 1093–1094
  - world knowledge acquisition, 1096
- Pollack, J., 204
- Polyhedra shape description, 835
- Polyphonic network, music and AI, 641
- Polytext, 460
- Polythetic hierarchy, 483
  - clustering technique, 107
- POM knowledge source, 75
- POPLOG system, 758
- Pop-out phenomenon, 116
- Popper, K. D., 223
  - logic of scientific discovery, 742
- Popperian inference method, 410, 415
- POP-11 language, 758
- POP-2 programming language, 758
- Popple, H. E., 440
- Portable Standard LISP, 524
- Position location, autonomous vehicles, 41–42
- Positive-horizon effect, 380–381
- Positive and negative instances, 466–469
- Possession theory, 838
- Possible world semantics:
  - belief systems, 64
    - partial-possible, 67–68
  - cross-world identification problem, 836
  - procedural semantics, 1030
  - propositional attitudes, 837
  - temporal reasoning, 873
- Postal address-reading machines, 86
- Potential fields, path planning and obstacle avoidance, 42, 713
- Potential function, pattern recognition, 723
- Power generality trade-off, 121
- Power relationships and office automation, 681
- PP-MEMORY system, conceptual dependency, 987
- Pragmatic grammars, natural-language interface, 656–657
- Pragmatics:
  - discourse understanding, 233, 241
  - hermeneutics, 366
- Prawitz, D., 132
  - theorem proving, 1117
- Precision in information retrieval, 420

- Predicate calculus:
  - automated programming, 18
  - commonsense reasoning, 833
  - frame theory, 308–310
  - machine translation, 565
  - planning, 749
  - procedural semantics, 1030
- Predicate logic, 538–543
  - computer-integrated manufacturing, 172
  - extensions, 541–543
    - descriptions, 541–543
    - identity, 541
  - introduction and elimination rules, 541–542
  - language, 538–541
    - deductive systems, 541
    - axiomatic, 541
    - expressibility, 540–541
    - semantics, 540
    - syntax, 539
  - limits of artificial intelligence, 495, 497–499
  - logic programming, 544
  - metatheoretic results, 543
  - processing, bottom-up and top-down
    - pattern matching, 784
  - second-order logic, 543
  - theorem proving, 1115
- PREDICT knowledge source, blackboard architecture, 74
- Predictability:
  - connectionism, 202–203
  - frame theory, 304, 308
  - future events, Bayesian decision methods, 52–53
- Predictive Analyzer, 780
- Predictive syntactic analysis, processing,
  - bottom-up and top-down, 784
- Preference semantics, 1025
  - machine translation, 570
- Prejudice, hermeneutics, 365
- Premises in logic, 536
- PRESS program, 549
- Price, K., 877–880
- Primal sketch:
  - depth map reconstruction, 1154–1155
  - early vision, 1131, 1142
- Primary colors, 125
- Primary memory, 276–277
- Primate visual systems, 1143–1144
  - parallel functional streams, 1143–1144
- Primitives, 759–761
  - actions:
    - planning, 748–749
    - postconditions, 749
    - preconditions, 749
- ACTs, 196–198
  - arguments against, 760
  - comprehensiveness, 759
  - defense of, 760–761
  - expectation-driven parsing, 697
  - finitude, 759
  - frame manipulation, 305
  - independence, 759
  - linguistic background, 759
  - machine translation, 568
  - natural-language generation, lexical choice, 649
  - pattern recognition, 725–726
  - semantic, 1025
- Principal variation search (PVS)
  - computer chess methods, 162
  - transposition tables, 165
- Priority queues, best-first search, 999
- PRISM language, language acquisition, 447
- Prisoner's Dilemma game, 615–616
- Probabilistic decision rule, Boltzman machine, 80
- Probabilistic inference, 419
- Probability:
  - Bayesian decision methods, 48–49
  - decision theory, 229
  - default reasoning, 841, 844–846
  - estimation, 845
  - medical advice systems, 586
- Problem-acquisition, 767–768
- Problem reduction, 762–766
  - applications, 766
  - divide-and-conquer technique, 763
  - heuristics learning, 474–475
  - pattern recognition, 726
  - representation, 762–763
  - search, 764–766, 995
    - algorithms, 765
    - backtracking, 46
    - heuristic search, 765
    - uninformed search, 765
- Problem solving, 767–778
  - anthropomorphic robots, 945
  - artificial intelligence and, 10–12
  - automatic programming synthesis, 21–22
  - blackboard architecture, 73–79
  - causal reasoning, 831–832
  - cluster analysis, 103
  - cognitive modeling, 114
    - behavioral data, 112
  - computational linguistics, 135
  - CONNIVER, 205
  - constraint satisfaction, 776–777
  - creativity, 223
  - declarative problem formulations, 767–768
  - dependency-directed backtracking, 47–48
  - derivation-formation spectrum, 771–772
  - distributed, 245–250
  - domain knowledge, 251
  - education applications of AI, 269
  - expert systems, 287
  - Fifth-Generation computers, 179–180
  - formation problems, 771–772
  - game playing, 318
  - General Problem Solver, 323
  - hybrid derivation and formation characteristics, 775–777
  - knowledge engineering, 288
  - learning, theory formation, analogy and abstraction, 15, 777–778
  - logic programming, 544
  - machine learning, 464–465, 473–476
  - major schemas, 772–775
  - means-ends analysis, 578–584
  - military applications, 606
    - specific projects, 608
  - natural-language interfaces, 655
  - objected-oriented programming, 452–453
  - optimization problems, 777
  - path planning and obstacle avoidance, 708
  - pattern matching, 719
  - pattern recognition, 720
  - phenomenology, critique of, 741
  - planning, relaxed reduction, 775
  - procedural problem formulations, 768–770
  - processing, bottom-up and top-down, 779
  - production schema, 772–773
    - move-selection function, 772
  - reasoning, 848
  - reduction schema, 774–775
    - divide-and-conquer method, 774
  - representation, 770, 777
  - search, 994
  - semantic networks, 1012
  - semantics, 1026
  - solution methods, 770–771
  - STRIPS system, 1099
  - STUDENT system, 1099
  - thinking and, 118
  - well-defined problems, 767
  - word-expert parsing, 707
  - see* Means-ends analysis
- Problem space:
  - machine learning, 467
  - search, 994–995
- Problem statement, feature correspondence, 621
- Procedural attachment:
  - inheritance hierarchy, 429
  - knowledge representation, 885
  - semantic networks, 1022
- Procedural interpretation, logic programming, 546
- Procedural knowledge, 113
- Procedural networks, natural language generation, 653
- Procedural representation, problem solving, 770
- Procedural semantic networks (PSNs), 1022, 1027–1030
  - computational linguistics, 136
  - KL-ONE program, 441
  - knowledge-base semantics, 1029
  - problems, 1027–1028
  - theoretical aspects, 1028
- Procedural/declarative controversy:
  - control structure, 214–215
  - inheritance hierarchy, 429
- Procedures:
  - coroutines, 219
  - problem solving, 768–770
- Process control, 96
- PROCESS markers, machine translation, 567–568
- Process planning:
  - robotics, 941
  - temporal, 873
- Processing, bottom-up and top-down, 779–784
  - acoustic-phonetic analysis, 1078–1079
  - Bayesian decision methods, 55
  - blackboard architecture, 74–75
  - branch-and-bound search, 1004
  - comparisons, 783–784
  - control structure, 214
  - data-driven, 781
  - early vision, 1137

- efficiency, 783–784
- examples, 781–783
- frame theory, 303
- goal-directed processing, 780
- grammars, 142
- heuristic search, 765
- history of terms, 780–781
- image understanding, 390
- knowledge system techniques, 289
- logic programming, 545
- natural-language, 660–661
- mixed strategies, 784
- multisensor integration, 636–637
- music applications, 641
- parsing, 781
- pattern matching and unification, 784
- pattern recognition, 725–726
- planning actions, 749
- rule-based systems, 781
- search, 781–783
- Processing elements, DADO, 227–228
- Processor scheduling, computer systems, 176
- PRODIGY system, 475
- Producer-consumer protocols, logic programming, 555
- Product decomposition, generalized cylinder representation, 322
- Production rules:
  - artificial intelligence, 12–14
  - heuristics, 379
  - linguistic theory, 444
  - logic programming, 552–553
  - medical advice systems, 585, 587
  - natural-language automatic programming, 31
  - pattern-directed retrieval, 718
  - pattern matching, 717, 719
  - problem reduction, 766
- Production systems, 823
  - architecture:
    - knowledge representation, 709–710
    - path planning and obstacle avoidance, 715
  - cognitive psychology, 116
  - connection machines, 200
  - control structures, 213
  - DADO, 227–228
  - forward and backward chaining, 780
  - inference, 418
  - knowledge engineering, 297
  - knowledge representation, 883, 888
  - speech understanding, 1079
- Production-type languages, ICAI, 157
- Program browser programming tool, 793
- Program derivation, logic programming, 550
- Program generation, theorem proving, 1122
- Program 17, self-reference, 1009
- Program synthesis:
  - pattern matching, 719
  - planning, 757
  - problem solving, 768
  - temporal reasoning, 870
- PROGRAMMAR system, SHRDLU and, 1048
- Programmed instruction, education applications, 267
- Programmer's Apprentice (PA), 787
- Programming:
  - automatic, 18–34
  - Horn-Clause logic, 546–547
  - logic, 544–557
    - databases, 551–552
    - forward *vs.* backward reasoning, 545
    - functional programming, 550–551
    - historical origins, 546
  - Horn-clause:
    - AND/OR nondeterminism, 546
    - extensions, 548–549
    - conditional subgoals, 549
    - metalevel subgoals, 549
    - negations, 548–549
    - procedural interpretation, 546
  - intelligent execution strategies, 554–555
    - loop detection, 555
    - subgoal selection, 554–555
  - LUSH corouting *vs.* PROLOG strategy, 546–547
  - object-oriented programming, 556–557
    - abstract data interpretation, 556–557
    - process data interpretation, 557
  - parallelism, 555–556
  - PROLOG, 547–548
  - query optimization, 552
  - recursive data structures, 547
    - minimal model semantics, 547
  - relational databases, 552
  - resolution, 545–546
  - rule-based expert systems, 552–554
    - declarative input-output, 553
    - heuristic *vs.* algorithmic programming, 553–554
  - specifications, 549–550
  - theorem-proving, 545
- proving properties, 1118
  - theorem proving, 1122
- robotics, 938–939
  - 3-D geometric models, 940–941
  - off-line programming, 940–941
  - simulation, 940–941
  - rule-based systems, 968–969
  - vs.* teaching computers, 491
- Programming assistants, 785–788
  - areas of knowledge, 786–787
  - comparisons, 788
  - experimental, 787–788
  - future trends, 788
  - programming environment, 795
    - in the large, 786
    - in the small, 785–786
    - tasks, 785–786
    - types of support, 786
- Programming environment, 789–796
  - AI programming systems, 789–791
  - AI tools and non-AI problems, 796
  - assistants, 785
  - automatic storage allocation, 791
  - commitment deferral, 791
  - conventional technology, 790–791
  - DWIM package, 795
  - extensibility, 795–796
  - garbage collection, 791
  - INTERLISP, 794–795
  - knowledge engineering, 296
- Programming environments, 789–796
  - languages, 791–792
  - programmer's assistant, 795
  - techniques, 792
  - tools, 792–794
    - contraction, 793
    - integration, 793
    - interactive interface, 792–793
    - knowledge-based, 793
  - uncertainty, 790
- Programming language:
  - design, control structures, 212–214
  - first-generation manipulator languages, 948
  - inheritance hierarchy, 422, 429–430
  - knowledge engineering, 296
  - limits of, 500–502
  - logic programming, 544
  - robotics, 938–940
    - flexibility and intelligence, 939
    - hardware dependency, 940
    - marketing, 940
    - system integration, 939
    - task-level commands, 939
    - user-language compatibility, 940
  - second-generation languages, robots, 948
- Programming Language for Natural-language Processing (PLNLP), 1022
- Progress in Machine Intelligence*, 721
- Progressive deepening, computer chess, 164
- PROLOG system, 796–797
  - artificial intelligence language, 15
  - associative memory, 18
  - ATN parser, 330
  - automatic programming synthesis, 22, 27–28
  - backtracking, 47
  - binary resolution, 893–894
  - computer architecture, 215
  - computer-integrated manufacturing, 172
  - computer systems, 174–175
    - storage management, 175
  - constraint satisfaction, 207, 210
  - control structures, 212, 214
    - logic programming, 215
  - coroutines, 222
  - DADO computer, 227–228
  - declarative programming, 269
  - definite clause grammar, 329–330, 339
  - domain knowledge, 252
  - GPSG parsers, 343
  - inductive inference search, 413–414
  - knowledge engineering, 296
  - knowledge representation, 882, 887
  - legal analysis, 460
  - limits, 500, 502
  - literature, 532–533
  - logic programming, 546
    - addition or deletion of clauses, 548
    - cut operator, 548
    - databases, 552
    - expert systems, 553
    - sequential control, 547–548
    - theorem-proving, 545–546
  - logical inference, 418
  - machine translation, 565
  - parsing, 142
    - left-corner, 691

- processing, bottom-up and top-down, 781
- pattern matching, 719
- problem reduction, 764
- programming environment, 790
- prostheses, 799
- question answering, 819
- reasoning, 823
- rule-based systems, 971
- semantic networks, implementation, 1022
- social issues of AI, 1050
- strategy, logic programming, 546–547
- Pronominal reference, semantic grammar, 352
- Pronouns:
  - discourse understanding, 236
  - natural-language interface, 658
- Proof:
  - completeness, 132
  - propositional logic, 561–562
- Proof-of-concept real-time scene analyzer, 801
- PROPEL:
  - expectation-driven parsing, 697
- Property inheritance, 422
  - artificial intelligence, 12
  - frame theory, 307
  - formalization, 309
- Property lists, LISP 1.5, 521
- Propositional attitudes, 837–838
- Propositional calculus, 558
  - law applications, 457
  - problem reduction, 766
  - processing, bottom-up and top-down, 780
- Propositional dependencies, belief reasoner, 60
- Propositional logic, 538, 558–563
  - artificial intelligence, 562
  - conjunctive normal form (CNF), 560
  - deductive systems, 561–563
    - semantics, 562–563
    - syntax, 561–562
  - disjunctive normal form (DNF), 560
  - language, 559–561
    - paradox of the material conditional, 561
    - semantics, 560–561
      - connectives, 560
    - syntax, 559–560
  - tautologies, contradictions and contingent propositions, 561
  - two-place truth-functional connectives, 559–561
  - nonmonotonic reasoning, 851
  - predicate logic, 538
  - primitives, 760
  - processing, bottom-up and top-down, 784
  - theorem proving, 1115–1117
- Propositional networks:
  - context enclosing, 1015
  - coreference links, 1015
  - semantic networks, 1014–1016
  - strict nesting, 1015
- PROSPECTOR system, 608, 797
  - Bayesian decision methods, 55
  - default reasoning, 844
  - domain knowledge, 252
  - knowledge representation, 889
  - propositional networks, 1016
  - rule-based systems, 971
  - uncertainty, Bayes' rule, 855
- Prostheses, 797–804
  - analytic methods, 799
  - historical context, 797–798
  - human-machine communication management, 799
  - human-machine symbiosis, 797
  - intelligent applications, 800–804
  - internal world model, 798
  - manipulation aids, 799
  - mobility aids, 798
  - omnidirectional mobility, 800
  - reading for visually impaired, 800–801
  - robotic manipulation aid, 801–802
  - self-expression and aesthetics, 799
  - sensory, 799
  - specific properties and design constraints, 799
  - task structure, 799
  - theoretical context, 798–799
  - user expertise, 799
  - user-machine-environment triad, 798
  - visual communication aids for language-impaired, 803–804
- PROTEAN system, knowledge engineering, 297
- PROTEUS program, 646–647
  - systemic grammar, 653
  - text generation, 144
- Protocol, 641
- Protocol analysis:
  - law applications, 461
  - medical advice, 584
- Prototype:
  - deformation model of conceptual structure, 459
  - frame theory, 302, 306–307
  - image understanding system, 391
  - military applications, 606
  - object-oriented programming:
    - knowledge sharing, 455
- PROUST programming tutor, 155
  - pattern matching, 719
  - programming assistants, 787
- Proximity sensing, 804–806
  - magnetic, 804
  - optic, 804–805
  - robot sensors, 1035
- PRUNE operator, beam search, 57
- Pruning:
  - alpha-beta, 4–6
  - branch-and-bound search, 1001–1003
  - forward:
    - checkers-playing programs, 90
    - computer chess, 162–163
    - limits of, 497–498
    - military applications, 606
- Pseudosensor whiskers, path planning and obstacle avoidance, 714
- PSI automatic programmer, 29–32
  - computational linguistics, 140
  - computer architecture, 216
  - LIBRA component, 785–786
- PSI-KLONE system, 1028
- PSL, 510
  - development, 525
  - history, 526
- PSN system, knowledge representation, 883, 885–886, 889
- Psychoanalysis, phenomenological hermeneutics, 367–368
- Psycholinguistics, 115, 118, 446–447, 506–507
  - natural-language generation, 653–654
- Psychology:
  - antimechanism and Godel argument, 739–740
  - cognitive, 115–118
  - computational theory of mind, 738
  - dot-pattern analysis, 255
  - early vision theory, 1142
  - experimental, 275
  - goal-directed processing, 780
  - intelligence and, 437
  - of memory, 278
  - naive, commonsense reasoning, 834
  - qualitative physics and, 808
  - semantic networks, 1012
  - social issues of AI, 1050
- PUFF system, 590
- PUGG system, story analysis, 1098
- Pull systems, manufacturing, 37
- PUMA arm, 927
  - industrial robots communication, 947
  - modular printer-carriage assembly, 938
- Pushdown store automata, ATNs, 327
- Push systems, manufacturing, 37
- Puzzle grammar (PG), 339, 342
- Pylyshyn, Z. W., 117–123, 284
- Pyramid models, image understanding, 397
- QED system, 94
- QLISP, pattern matching, 717, 719
- Quadtrees device:
  - segmentation, 879
  - sensor mapping, 41
  - volumetric description, 863–865
- Qualitative description:
  - education applications, 268–269
  - scale-space methods, 973–974
- Qualitative laws for discovery, 483–484
- Qualitative physics, 747, 807–813
  - ambiguities, 810
  - analysis, 808
    - component-based, 808–811
    - constraint-based, 808–809
    - process-based, 808–811
  - arithmetic, 809
  - calculus, 811
  - confluences, 809
  - current and future research, 812–813
  - derivatives, 809
  - envisioning, 811–812
  - literature survey, 813
  - modeling, 810
  - psychology and, 808
  - quantity spaces, 809–810
  - structure and function, 808
  - temporal reasoning, 871–873
- Qualitative reasoning:
  - belief revision, 61
  - temporal reasoning, 872
- Qualitative shape representation, 863

- Quantification:
  - natural-language interface, 658
  - procedural semantics, 1030
- Quantifiers:
  - inheritance hierarchy, 426
  - predicate logic, 538, 540
- Quantitative laws for discovery, 484–485
- Quantitative theory of computational complexity, 491
- Quantities in commonsense reasoning, 834–835
- Quasi-indicators, belief systems, 66
- Qubic, language:
  - deep five force, 318
  - search reduction, 315
- Query-the-User reasoning, 819
- Question answering, 814–821
  - answers, 815–817
    - correct answers, 815–816
    - intensional knowledge, 816
    - nonmisleading answers, 816–817
    - useful answers, 816
  - augmented transition networks, 325
  - goal-related information:
    - appropriate plans, 817
    - inappropriate plans, 818–819
  - indirect answers, 817
  - information to clarify, 820
  - information to justify or explain, 819
  - logic databases, 551–552
  - memory organization packets, 592
  - misconceptions, 820–821
  - misconstruals, 821
  - procedural semantics, 1029
  - questions, 814–815
  - rejecting the question, 821
  - responses, 817
  - SAM system, 986
  - script activities, 991
  - semantic memory, 593
  - semantic networks, 1012
  - story analysis, 1091
    - script application, 1093
  - theorem proving, 1118, 1121–1122
  - theoretical framework, 814
- Quiescence search:
  - computer chess methods, 163
  - horizon effect, 380
- Quillian memory models, 593
  - connectionism, 204
- Q-32 computer, LISP development, 522
  
- Radar systems and pattern recognition, 728
- RAIL language, 948
- Random-dot stereograms, 1084
- Randomness and uncertainty, 854
- RAND system, distributed problem solving, 250
- Range sensing:
  - problems, 932–933
  - robot sensors, 931, 1034–1035
- RANSAC system:
  - motion analysis, 629
  - stereo vision, 1085, 1089
- Rapaport, W. J., 538–543, 558–563
- Rapid generalization, concept learning, 191
- RAPT language for industrial robots, 951
- Razoring in computer chess, 162–163
- Reaction time studies and semantic memory, 593
- Reactivation point in coroutines, 220
- Reading comprehension and cognitive modeling, 111, 113–114
- Reading machines, 86–87, 800–801
- Realization:
  - Bossie component, 651
  - natural-language generation, 644–645
- Reasoning, 822–826
  - AND/OR graphs, 7
  - artificial intelligence and, 11
  - autoepistemic, 851
  - backward and forward, 11–12
    - processing, bottom-up and top-down, 779
  - belief systems, 63
  - binary resolution, 892
  - biological systems, 924
  - blackboard architecture, 74
  - case-based, 477–478
  - causal, 827–832
    - inference, 418–419
    - medical advice system, 584–585
    - primitives, 761
    - theorem proving, 1117
  - cognitive modeling, 111, 114
  - commonsense, 833–838
    - hermeneutics, 362
    - legal analysis programs, 459
  - default, 12, 840–846
  - epistemology, 285
  - evidential, 858
  - explaining process, 600
  - focus of attention, 848–849
  - formal logic, 848
  - forward:
    - processing, bottom-up and top-down, 779
  - game-playing programs, checkers, 88
  - general-purpose *vs.* special-purpose system, 823
  - goal-directed, 12
  - image understanding, 393
  - inference applications, 418, 824–825
    - knowledge representation, 826
  - information retrieval, 433
  - knowledge sources and, 280
  - limits of artificial intelligence, 495
  - logic and, 537
  - meta-reasoning, 598
  - methodology classification, 822–823
  - naive physics, 747
  - natural language, 649, 848–849
  - natural-language interfaces, 655
  - nonlinguistic, natural-language generation, 644–645
  - nonmonotonic, 849–852
  - physical, 869
  - planning process, 599–600
  - plausible, 854–862
    - knowledge systems, 288
    - see also* Uncertainty
  - precise *vs.* imprecise, 823
  - problem level *vs.* meta-level, 823
  - qualitative physics, 807
  - question answering, 815
- resource-limited, 848–849
- rule selection, 826
- semantic networks, 1012
- spatial, 863–869
  - physical behavior, 836
- subgoal solution, 826
- systems, 824–826
- temporal, 870–874
- theorem proving, 1115
- thinking and, 118
  - see also* specific types of reasoning
- Reason-maintenance programs, commonsense reasoning, 834
- Recall:
  - episodic memory, 275–279
  - information retrieval, 420
- Receiver-sender communications, 221
- Recency effect, memory property, 277–278
- Recognition:
  - episodic memory, 275–279
  - frame theory, 304
    - memory organization, 305–306
- Recognition cones, image understanding, 397
- Recollection:
  - all-or-none model, 278–279
  - encoding-specificity principle, 278–279
  - episodic memory, 275–279
  - generate-recognize model, 278–279
  - threshold model, 278–279
- Recurrence relations, LISP synthesis, 26
- Recursion, 875–876
  - Church's thesis, 99
  - cybernetics, 226
  - limits of, 876
    - natural-language generation, 653
- Recursive-descent parsing, 689–690
- Recursive-function theory, self-replication, 1011
- Recursive transition networks (RTNs), 324–325
  - context-free grammars, 326–327
- Earley algorithm, 330
- history, 325–326
- inductive inference, 411
- Reddy, D., 362
- Reddy, R., 362
- Reduced-instruction-set computer (RISC), 176
- Reduce goal, means-ends analysis, 579
- REDUCE system, 96
  - LISP dialects, 524
- REF-ARF programs, 876–877
- Reference beacons, autonomous vehicles, 41
- Reference trajectory, robot control systems, 906
- REFINE-DESIGN knowledge system, 77
- Refinement relation, inductive inference search, 413–414
- Reflectance map, shape analysis, 1042
- Reflection, control structure, 213
- Reflective systems, self-reference, 1009
- Reflexive-deletion paradigm in case grammar, 336
- Reformulation and problem solving, 770
- Refutation tables:
  - completeness, 132

- computer chess, 164–165
  - software advances, 166–167
- Region-based segmentation, 877–880
  - combinations, 879
  - early vision, 1133
  - evaluations, 880
  - region merging, 877
  - region splitting, 877–879
  - shape analysis, 1047
  - split and merge techniques, 879
- Region growing, color vision, 129
- Region splitting, 877–879
  - color vision, 128–129
  - histogram-based, 878–879
- Regular sets, augmented transition networks (ATNs), 326
- Regularization theory:
  - intensity changes, 258, 265–266
  - shape analysis, 1043
- Reification of position, temporal reasoning, 870–872
- Reinefeld's depth-2 idea, computer chess, 162
- Reinforcement learning, demons, 232
- Reinterpretation, frame theory, 304
- Reiter, R., 309
  - default reasoning logic, 842–843, 851
- REL (Rapidly Extensible Language) system, computational linguistics, 135, 138
- Relational databases, 552
  - assertions, 289
- Relational grammar, 354
- Relational graphs:
  - semantic networks, 1013
  - verb-centered, 1013–1014
- Relations, inheritable, 428
- RELATUS Natural-language system, 373
- Relaxation techniques:
  - algorithms
    - constraint satisfaction, 209–210
    - feature extraction, 301
  - labeling:
    - early vision, 1143
    - constraint satisfaction, 1140
  - segmentation, 879
    - color vision, 130
  - word recognition system, 87
- Relevance feedback:
  - belief revision, 60
  - discourse understanding, 241
  - information retrieval, 420–421
  - system, 537
- Reliability, inductive inference, 415
- Reminding phenomenon, frame recognition, 306
- Remote center compliance, robotics, 927
- Remotely-piloted vehicles (RPVs), 39
- Remote sensing, early vision, 1131
- RENDEZVOUS system, database interface, 138
- Rennel, R., 362
- Rennels, G., 584–590
- Repair robots, 957
- Representation:
  - analogue, 881
  - basic model, 201–202
  - binary resolution, 895
  - checkers-playing programs, 89–90
  - cognitive science, 123
  - computational theory of mind, 738
  - conceptual dependency, 194
  - connectionism, distributed, 203
  - control structure, 214–215
  - epistemology, 281
  - generalized cylinders, 321–323
  - inheritance, 423
  - knowledge, 882–889
    - cognitive modeling, 111–112
    - see also Knowledge representation
  - logical template, 459
  - machine learning, 473
  - natural-language generation
    - intermediate level, 651–652
  - predicate logic, 538
  - problem reduction, 762–763
  - problem solving, 770
  - propositional logic, 558
  - skeletal, 84
  - speech-act theory, 1064–1065
  - speech understanding, 1079–1080
  - wire-frame, 890–892
  - word-expert parsing, 703
- REQUEST system, database interface, 138
- Requirements analysis, programming assistants, 786
- RESEARCHER system, 421, 592
- Resolution:
  - automatic programming deductive mechanism, 20
  - binary, 892–901
  - commonsense reasoning, 833–834
  - completeness, 132
  - computer systems, 174
  - definite-clause grammar, 341
  - inference, 418
  - logic programming, 537, 545–546
  - predicate logic, 498, 541
  - theorem proving, 1117
- Resource limitations:
  - reasoning, 848–849
  - expert systems, 849
- RESUME action, ATN grammar, 328–329
- RESUMETAG, ATN grammar, 328
- Retargeting, d-node, 1000
- Reverse transformational grammar, 360
- Revised therapy algorithm, 589–590
- Rewriting:
  - natural-language generation, 650
  - theorem proving, 1118–1119
- Rich, E., 9–16, 309
- Ricoeur:
  - phenomenological hermeneutics, 367–369
  - theory of interpretation, 368–369
- Rieger's inference model, 1091
- Riesbeck, C. K., 696–701
- Rigid-body models, robot-control systems, 909–911
- Rigidity assumption, optical flow, 686
- Risk magnitude, medical advice systems, 589
- Rissland, E. L., 459–460
- RITA rule-based systems, 971
- RLISP, 510–511
- RLL system, 476
  - domain knowledge, 252
  - knowledge engineering, 296
- Road detection, autonomous vehicles, 40–41
- Road following:
  - mobile robots, 959
  - path planning and obstacle avoidance, 713–714
- Roberts, R. B., 312
- Robinson, A., completeness, 132
- Robinson, J., theorem proving, 1117
- Robinson's resolution:
  - logic programming, 546
- ROBOT system:
  - computational linguistics, 139, 141
  - semantic grammar, 350
  - tactile sensors, 493
  - Tower-of-Hanoi puzzle, 23–24
- Robot-control systems, 902–921
  - appendix of terms, 921
  - n-degree-of-freedom rigid-body model, 913
  - general robot-arm dynamics, 902, 909–913
    - feedback control, 913–916
      - task-encoding methodology, 913–916
    - gradient vector fields and Hamiltonian systems, 914–915
  - local nonlinearities, 912–913
  - rigid-body model, 909–913
    - kinematics, 910
    - omissions, 911–913
    - transformations and frames of reference, 909–910
  - servo control, 916–921
    - adaptive gravity cancellation, 921
    - adaptive torque, 920–921
    - coordinate transformation schemes, 918–919
    - global adaptive controllers, 920–921
    - linear approximations, 919–920
    - linearization by coordinate transformation, 916–919
    - local adaptive and learning techniques, 919–920
    - robust cancellation via sliding modes, 917–918
    - task-encoding methodology, 913–914
  - single-degree-of-freedom robot arm, 903–909
    - dynamics, 903–904
    - feedback control, 904–909
      - robust properties, 905
      - transient response, 905–906
- Newtonian dynamic model, 903
- servo problem, 906–909, 917
  - adaptive control, 908–909
  - forced response of linear systems, 905–906
  - inverse dynamics, 907
  - robust tracking, 907–908
- Robotic Industries Association (RIA), 945
- Robotics, 923–941
  - adaptive control, 937–938
    - arc welding, 938
    - compressor-cover assembly, 937
    - modular printer-carriage assembly, 938
  - tracking, 937–938
- autonomous planning, 802
- biological systems, 924
- capability, components and intelligence, 923
- circumscription, 102
- classification, 924



- commands, 802
  - contact sensing, 935–936
  - control systems, 802, 902
  - development incentives, 924–925
    - limitation, 924
  - dialogue management, 802
  - direct range measurement, 930–932
  - early vision, 1131
  - end-effectors, 927
    - hand, 927
    - hand-tool holder, 927
    - micromanipulator, 927
    - tool, 927
  - feedback control of robot arms, 902
  - first-generation, 802
  - force sensors, 937
  - grasp automation, 802
  - industrial, *see* Industrial
  - manipulation, 803, 925
    - joint-to-world coordinate transformation, 925
  - manufacturing processing planning, 941
  - mobility, 803, 927–928
    - control, 928
    - surfaces and locomotion, 927–928
  - motion trajectories, 925–926
    - control, 926
    - dynamics, 925–926
    - smooth path, 925
    - work-station transforms, 925
  - noncontact sensing, 929–930
  - tactile sensors and, 936–937
  - programming, 938–941
    - reflexes, 803
  - prostheses and, 797–804
    - manipulation aid, 801–802
  - proximity sensing, 804–806
  - range image processing, 933–935
    - faces, 933–934
  - research and development, 925, 935
    - tactile sensors, 937
  - robot characteristics, 923–925
  - second-generation, 802
  - sensing, 39, 928–929
    - one-dimensional, 39
    - sequence, 928–929
    - signals, 929
    - strategy, 929
    - two-dimensional, 39
    - three-dimensional, 39
  - sensors, 1031–1036
  - six-legged vehicles, 928
  - socioeconomic problems, 924–925
  - spatial knowledge and inference, 835
  - tactile sensors, 935–937
    - noncontact sensing, 936–937
    - transducers, 935
  - technical approach, 925
  - teleoperators, 1100–1101
  - volumetric representation, 935
- Robots:**
- anthropomorphic, 944–954
  - computer-integrated manufacturing, 173
  - configuration space, 711
  - contemporary, 945–949
  - functionalist critique, 738
  - industrial:
    - taxonomy, 945–947
    - see also* Industrial robotics
  - language acquisition, 443
  - manipulators, 571–575
    - trajectory planning, 574–575
  - military, 954
  - mobile, 957–961
    - limits of, 496–497
    - path planning and obstacle avoidance, 711
  - motion analysis, 620
  - multisensor integration, 632–637
  - path planning and obstacle avoidance, 708
  - pattern recognition, 720–721
  - personal, 954
  - planning, 748
  - social issues of AI, 1051–1052
  - taxonomy, 945–948
- Robust parsing and natural-language understanding, 671–672**
- Rochester, Nathaniel, 520
- R1 systems, 10
  - knowledge engineering, 288, 297
- RONY problem, 476–477
- Rosch, E., 117, 306
- Rosenberg, J., 99, 441, 564, 758, 848–849, 1039
- Rosenblatt, Frank, 204
  - perceptrons, 729–730
- Rosenbloom, P., 998–999
- Rosenblueth, A., 226
- Rosenkrantz equivalent grammar, 690–691
- ROSE system, 963–964
- ROSIE system, 296
  - domain knowledge, 252
  - rule-based systems, 971
- Rote learning, checkers-playing programs, 90
- Route finding:
  - mobile robots, 959
    - obstacle avoidance, 959
    - preprogrammed path guidance, 959
    - road following, 959
    - structure following, 959
  - spatial reasoning, 866–867
- Route planning, 755
  - mobile robots, 959–960
- Router network, connection machines, 199
- Route search, autonomous vehicles, 43
- RS-232 ports, industrial robots, 947
- Rubber sheet, template matching, 576
- Rubik's cube:
  - means-ends analysis, 578
  - search methods, 994–995
- Rule-based systems, 963–973
  - applications, 964–965
  - architecture, 968
  - artificial intelligence and, 967
  - cognitive modeling, 114
  - cognitive psychology, 116
  - cognitive science, 123
  - conceptual evolution, 969–970
  - domain knowledge, 251
  - education applications of AI, 268
  - EMYCIN, 275
  - evolutionary system development, 968
  - forward and backward chaining, 780
  - heuristics, 379–380
  - image understanding, 398–399
  - implementation and availability, 972
  - legal analysis programs, 460
  - limits of, 497, 500
  - list of specific system, 964
  - logic programming, 552–554
  - medical advice systems, 454, 587
  - overview, 965–967
  - problem reduction, 766
  - processing, bottom-up and top-down, 779, 781
  - programming, 968–969
    - limits, 500
  - programming environment, 792
  - reputation *vs.* reality, 972–973
  - search and, 968
  - self-reference, 1006
  - speech understanding, 1079
  - techniques, 288–289
  - technology evolution, 971
  - transformational grammar, 356
  - word-expert parsing, 707
  - XCON system, 1166
- Rule-directed vehicle pilot, 42
- Rule-representation language and uncertainty, 854
- RULEGEN subprogram, 190–191
- RULEMOD subprogram, 190–191
- Rules:
  - error diagnosis, 154
  - grammatical, linguistic competence and performance, 504
  - melodic and harmonic, AI in music, 639
  - tutoring, ICAI programs, 154
- Rumelhart, D., 204, 446
- RUMMAGE system, 107
  - conceptual clustering, 483
- Run time typing, 517–518
  - Symbolics 3600, 177
- RUP system, belief revision, 58
- RUS system, natural language understanding, 668
- Russell, Bertrand:
  - predicate logic, 542
  - self-reference, 1005
- Russell's paradox, 543
- Russell, Stephen, 521
- Ruzzo chart parser, 694
- RX project, 831
- S.1 expert system, 290–291
- S-1 LISP, 525–526
- S-1 Mark IIA supercomputer, 525
- S-432, rule-based systems, 971
- Sabatier, P., 339
- Sabbah, D., 204
- Sacerdoti, E., 677
  - ABSTRIPS, 582
- Sachs, J., 448
- SAD SAM system, computational linguistics, 134–135
- SAFE synthesizer, natural language program synthesis, 32
- SAGE.2 system, 475
- Sail language, control structure, 213
- SAINT (Symbolic Automatic INTEgrator), 718, 973
- SAM system, 973
  - discourse understanding, 239–240
  - expectation-driven parsing, 697–698, 701

- frame theory, 304
- scripts, 981
  - natural language understanding, 985–987
  - pattern matching, predicting and instantiating, 987
- story analysis, 1092
  - world knowledge acquisition, 1096–1097
- theorem proving, 1117
- Sam's lemma:
  - binary resolution, 900
  - theorem proving, 1117
- Samuel, A. L.:
  - checkers-playing programs, 89–92
  - learning, 317
  - concept learning, 186
  - expert systems, 733
  - heuristics learning, 474
- Sapir-Whorf hypothesis, hermeneutics, 365
- SBT parsing, 690
- Scalar implicatures, 144
- Scale-space imaging, 392, 973–979
  - coincidence assumption, 974
  - definition, 975
  - edge detection, 1135
  - identity and localized assumptions, 976–977
  - image tree, 977–978
  - interval constraint, 978
  - Mandelbrot's view, 974–975
  - organization, 977–978
  - qualitative description and problem of scale, 973–975
  - qualitative structure, 975–976
  - raw description, 976–977
  - stability criterion, 978
  - zero crossings, 261–262
    - two-dimensional, 264–265
- Scale-space representation, 973–979
- SCARA (Selective Compliance Assembly Robot Arm), 945, 947
- Scene Analysis Program, 389, 732
- Scenes:
  - memory organization packets, 591–592
  - script structure, 983
- Schank, R., 233–244
  - cognitive modeling, 145
  - conceptual dependency, 495–496
  - conceptual dependency, 194
  - deep case grammar system, 337–338
  - memory organization packets, 591
  - frame theory, 304
  - story understanding, 828
- SCHED system, office automation, 140
- Scheduling procedures:
  - blackboard architecture, 78–79
  - blackboard systems, 79
  - CRYALIS system, 77
- Schema concept:
  - circumscription, knowledge representation, 889
  - cognitive psychology, 116
  - constraint satisfaction, 209
  - frame theory, 302
  - history, 302
  - script acquisition, 991–992
- Schema Representation Language
  - computer-integrated manufacturing, 173
- Schemata:
  - commonsense reasoning and frame theory, 834
  - data-driven processing, 781
  - episodic memory, 279
- Scheme dialect:
  - CommonLisp, 526
  - control structure, 212
  - virtual machine generalization, 213–214
- development, 525
- lambda calculus, 442
- LISP system, 512
- Schenker, D., 641
- Schleiermacher, hermeneutics, 363–364
- Schneider, D., 371
- SCHOLAR system, 980
  - computational linguistics, 139–140
  - human-computer interaction, 385
  - ICAI, 154–155
  - semantic memory, 593
- Schonfinkel, 441
- Schubert, L. K., 594
- Schwartz, J., 488–501
- Science:
  - computers in education and, 183
  - logic of discovery, 742
  - knowledge, 283
  - pattern recognition, 728
- Scientific community metaphor, distributed
  - problem solving, 249
- Scientific Datalink, 530
- Scientific Personal Integrated Computing Environment (SPICE) project, 526
- Scientific text processing, computational
  - linguistics, 141
- SCORE operator, beam search, 57
- Score printing and reading, AI in music, 638
- Scoring hypotheses, inexact inference, 586
- SCOUT system, branching factor, 82
- Script application:
  - story analysis, 1092
- Script headers, 982–983
  - direct header (DH), 982–983
  - instrumental header (IH), 982
  - locale header (LH), 982
  - precondition header (PH), 982
- Scripts, 980–992
  - acquisition, 991–992
  - applications, 983–984
  - cognitive psychology, 116
  - discourse understanding, 233, 249–250
  - domain knowledge, 251–252
  - episodic memory, 279
  - frame theory, 302
    - story processing, 304
  - human-computer interaction, 384
  - inferences, 984
  - information retrieval, 421
  - limits of artificial intelligence, 496
  - memory organization packets, 591
  - natural language understanding, 985–990
    - argument units, 990
    - BORIS system, 988–989
    - FRUMP, 988
    - OpEd system, 990
    - SAM, 985–986
- object-oriented programming, 452
- path information, 984–985
- personal relations, 838
- phenomenology and, 741–742
- primitives, 759
- problems, 985–986
- pronoun resolution, 984
- psychological validity, 990–991
- reasoning, 826
- semantically-based
  - human-computer interaction, 384
- stereotyped social situations, 838
- story analysis, 304
  - world knowledge, 1092–1093
- structure, 981–983
  - events, episodes and scenes, 983
  - invocation with script headers, 982–983
- visual communication aids, 804
- word-sense disambiguation, 984
- SEAC system, 95
- Search, 994–998
  - activation, image understanding, 392
  - algorithm, inheritance hierarchy, 424
  - AND/OR graphs, 7–8
  - artificial intelligence and, 10–12
  - backtracking, 46
  - backward, processing, bottom-up and top-down, 783
  - beam, 56–57, 998
  - best-first, 997–999
  - bidirectional, 996, 1000
    - problem solving, 773
  - processing, bottom-up and top-down, 784
    - semantic memory, 593
  - blackboard architecture, 73
  - branch-and-bound, 1000–1004
  - branching factor, 81–82
  - breadth-first, 547, 995–996
    - medical advice systems, 585
    - reasoning, 825
  - brute-force, 995–996
  - checkers-playing programs, 90
    - parallel trees, 92–93
  - combined approaches to, 469–470
  - computer chess, exhaustive and selective, 165–166
  - constraint satisfaction, 206
  - control structure, 211
  - depth-first, 996, 1004–1005
    - alpha-beta pruning, 4–6
    - coroutining, 221
    - iterative-deepening, 996
    - knowledge representation, 887
    - machine learning, 467
    - problem reduction, 764
    - reasoning, 825
  - early vision, relaxation labeling, 1140
  - errors, computer chess, 169
  - failure-directed, 396
  - forward and backward, 1000
    - processing, bottom-up and top-down, 783
  - game playing, 315–316
  - game trees, 321
  - general-to-specific methods, 468–469
  - goals, creativity, 224

- heuristic, 996–997
  - REF-ARF program, 876–877
- hill-climbing, 997
- inductive inference, 412–413
- information retrieval, techniques, 420–421
- inheritance hierarchies, 423–424
- iterative-deepening A\* (IDA\*), 998
- knowledge-system techniques, 288
- local neighborhood, checkers-playing programs, 91
- machine learning, 464, 466–467
- minimax procedure, 614
- multisensor integration, 636
- parallelism, 556
- parsing and, 142
- path planning, 710
  - mobile robots, 960
- planning and operation, 750
- problem reduction, 764–766
  - algorithms, 765
  - game playing, 312
  - graph, 764
  - heuristic search, 765
  - uninformed search, 765
- problem solving, 772–773
  - production schema, 772
- problem spaces, 994–995
  - graph representation, 994–995
- processing, bottom-up and top-down, 779, 781–783
- reasoning:
  - AND/OR representations, 825
  - game-playing, 848
- recursion and, 875
- rule-based systems and, 968
- selective, computer chess, 168
- self-reference, 1005
- set, associative memory, 17
- space, 56
  - modified constraint, 46
  - problem formulation, 46
- specific-to-general methods, 467–469
- speech understanding processing strategies, 1080
- state space, *see* State space search
- stereo vision, coarse-to-fine strategies, 1088
- tree, 468–469
  - Waltz filtering, 1162
- version-space method, 469–470
- see also* types of Search, e.g., Depth-first search
- Searle, 1053
- Second-order logic, 538, 543
- SEEK program, 586
- SEER system, image understanding, 396–397
- SEG knowledge source, 75
- Segmentation:
  - early vision, 1132–1133
    - description, 1139
    - surfaces, 1141–1142
  - edge detection and, 1133
  - region-based, 877–880
    - combinations, 879
    - evaluations, 880
    - region merging, 877
  - region splitting, 877–879
    - split and merge techniques, 879
  - scale-space imaging, interval tree, 978
- Self-consciousness, hermeneutics, 365
- Self-organization:
  - limits of, 491
  - systems in cybernetics, 226
- Self-reference, 1005–1010
  - autonymy, 1008
  - consciousness, 1009–1010
  - cybernetics, 226
  - indexicality, 1007–1008
  - introspection, 1008–1009
  - metalevel reasoning, 1005–1006
  - nonmonotonic reasoning, 850
  - reflection, 1009
  - varieties, 1007–1009
- Self-replication, 1010–1011
- Selfridge, M., 447
  - demons, 232
- Selfridge, O. G., PANDEMONIUM system, 687
- Self-understanding systems, 1005
- Semantic checking:
  - case frames, 334
  - case grammar, 338
- Semantic dependency tree, 145
- Semantic distance, 593
- Semantic grammar, 331–332, 350–353
  - advantages, 351–353
  - discourse phenomena, 352
  - efficiency, 351–352
  - habitability, 352
  - limitations, 353
  - natural-language generation, 647
  - natural-language interface, 656–657
  - natural language understanding, 667
- Semantic Information*, 3
- Semantic interpretation:
  - linguistic competence and performance, 503
  - transformational grammar, 357
- Semantic markers:
  - machine translation, 567–568
  - primitives, 759
- Semantic memory, 593–594
  - direct extensions, 593
  - episodic memory, 275–276
  - Quillian's models, 593
- Semantic net:
  - commonsense reasoning, 834
  - natural-language generation, 652
  - reasoning, 826
- Semantic networks, 1011–1023
  - analogue representation, 881
  - artificial intelligence, 12
  - belief spaces, 65
  - case frames, 335
  - character recognition, 83–85
  - cognitive modeling, 111, 113
  - cognitive psychology, 116
    - memory, 117
  - computational linguistics, 136–137
    - computer-aided instruction, 139–140
  - concept learning, 186
  - connection machines, 200
  - connectionism, applications, 204
  - deep case grammar, 335
  - domain knowledge, 251
- generic and individual concepts, 1020
- history, 1012–1013
- image understanding, 398–399
- implementation level, 1022–1023
- information retrieval, 421
- inheritance hierarchies, 422, 1017–1018
- intensions and extensions, 1020–1021
- knowledge representation, 882, 885
  - limits of, 500
- knowledge system construction, 294
- language acquisition, sentence-meaning pairs, 480–481
- legal applications, 460
- limits of artificial intelligence, 495
- logic, 1018–1019
- machine learning, 1021–1022
- medical advice systems, 585
- memory, 593–594
- parallel-marker propagation, 429
- parsing and generation, 1021
- pattern matching, 717
- procedural attachments, 1022
- propositional networks, 1014–1016
- PSI automatic programmer, 29–31
- relational graphics, 1013
- taxonomic and assertional systems, 1019–1020
- type hierarchy, 1016–1017
- verb-centered relational graphs, 1013–1014
- Semantic Representation Language (SRL)
  - parsing techniques, 1021
- Semantics, 1024–1029
  - belief systems, 63
  - computational linguistics, 143
  - conceptual dependency, 195
  - coroutine implementation, 220
  - decompositional, 1025–1026
  - epistemology, 284
  - extensional properties, 540
  - hermeneutics, 366
  - intensional properties, 540
  - knowledge representation, 883–884
  - knowledge-base, 1028–1029
  - lexical-interpretive, 373
  - limits of artificial intelligence, 495
  - logic and, 536–537
  - logic programming, 547
  - model-theoretic, 760
  - Montague grammar, 1026
  - natural language, 495
    - interface, 657
  - predicate logic, 540
  - procedural, 1027–1030
  - programming languages, limits of, 500
  - propositional logic, 559–561
    - deductive systems, 562–563
  - situation, 1026–1027
  - speech understanding, 1077
  - syntax and, 1024–1025
  - theories, 1025
  - Turing test, 1129
- Semiotics and hermeneutics, 372
- Sense-discrimination language (SDL), 703–704
- Sensitivity analyses, decision theory, 229
- Sensor-controlled automation (SCA), robotics, 950

## Sensors:

- industrial automation, 39
  - industrial robots, 946, 951–952
  - information:
    - autonomous vehicles, 41
    - military applications, 606–609
  - multisensor integration, 632
  - robots, 1031–1036
    - ACTIVE, 1032–1034
    - CONTACT devices, 1033
    - contact sensing, 1035
    - direct/indirect, 1031
    - global/local, 1031–1032
    - internal/external, 1031
    - INVESTIGATIVE, 1032–1034
    - mobile robots, route planning, 960
    - PASSIVE, 1032–1034
    - proximity sensing, 1035
    - range sensing, 1034
    - SAMPLE, 1032–1033
    - static/dynamic, 1032
    - tactile sensing, 1035
    - vision, 1033–1034
  - taxonomy, 1031–1033
  - uncertainty, 41
- Sensory aids for prostheses, 797–798
- Sensory functions, 493–496
- image analysis, 493
  - natural language analysis, 494–496
  - recognition of speech, 493–494
- Sentence structure, ellipsis, 273
- Sentence understanding:
  - case frames, 334
  - case grammar, 333
- Sentential logic, 558
- Serum Protein Diagnostic Program, 585, 590
- Servo problem, robot control systems, 906–909
- SETF macro, 514
- SETL programming assistant, 788
- limits of, 502
- Set search, associative memory, 17
- Set-of-support strategy, theorem proving, 1118
- Set of Support Theorem, 899
- S-expression
  - LISP, 510–511, 521
  - LISP, 1.5, 521
- Shading from shape, early vision, 1141
- Shafer, J., region splitting, 878
- Shafer, S. A., 130
- Shakey Robot Project, 9, 1039
- anthropomorphic robots, 945
  - autonomous vehicles, 39
  - obstacle detection, 40
  - control, 928
  - dead reckoning position location, 42
  - navigation, 953
  - path planning and obstacle avoidance, 708
  - knowledge representation, 709–710
- Shannon, C., 380
- computer chess, 159
- Shape analysis, 1039–1047
- commonsense reasoning, 835
  - computational task, 1040–1041
  - computer-aided design (CAD), 950
  - criteria, 1040
  - early vision, 1141
  - shape from shading, 1041–1044
  - texture analysis and, 1110–1111
  - three-dimensional, 1046–1047
    - extended Gaussian image, 1046–1047
    - generalized cylinders, 1046
  - two-dimensional, 1044–1046
    - global properties of binary images, 1044
    - normalized texture property map, 1046
    - skewed symmetry, 1045–1046
  - visible surfaces, 1044–1045
- Shape from shading, 204
- binocular stereo, 1044
  - constraint satisfaction, 210
  - early vision, 1141
  - image irradiance equation, 1041
  - intensity discontinuities, 1145
  - occluding contour, 1042–1043
  - photometric stereo, 1043–1044
  - qualitative, 1144
  - shape analysis, 1041–1044
- Shape-from-X theory, 1147
- early vision, 1141
- Shape matching, Hough transform, 383
- Shape recognition:
  - geometric properties, 301
  - moment invariants, 301
- Shape representation:
  - boundary descriptions, 864
  - part-whole description, 863
  - physical reasoning, 869
  - qualitative, 863–869
    - discontinuities, 865
    - symbolic vocabulary, 864
  - volumetric descriptions, 863–864
- Shapiro, S. C., 799–784
- Shapiro synthesis algorithm, 27–28
- Shastri, L., 203–204
- Shaw, D. E., 678–679
- Shaw, J. C., 9
- game playing, 316
  - general problem solver (GPS), 323
- Sheil, B., 791–796
- Shepard, R., 117
- Shirai, Y., 804–806
- image understanding, 394–395
- Sholam, Y., 870–874
- Shortfall scoring, speech understanding, 1080–1081
- Shortliffe, E. H., 10, 584–590
- SHRDLU system, 1048
- computational linguistics, 137
  - grammar formalism, 143
  - hermeneutics and, 371–372
  - history, 10
  - human-computer interaction, 384
  - industrial robots, 950
  - MICRO-PLANNER, 604
  - natural-language generation, 645–647
  - pattern matching, 719
  - phenomenology, 732
  - PLANNER and, 748
  - script applications, 983–984
  - self-reference, 1006
  - text generation, 144–145
- Shumaker, Randall, 604–613
- SIAP (Surveillance Integration Automation Project), 607–608
- knowledge engineering, 297
- SIGMA system, image understanding, 399–400, 402
- Sign preservation, robot control, 918
- Signal processing:
  - pattern recognition, 721
  - Viterbi algorithm, 1160
- Signature tables in checkers-playing, 91–92
- Sigurd's program, natural-language generation, 645–646
- SIMD (single-instruction multiple-data machine):
  - associative memory, 17
  - CM, 216
- Similarities and intelligence, 432
- SIMILARITY system, failure-directed activation, 396
- Simon, H. A., 9, 302, 1052–1053
- game playing, 316
  - general problem solver (GPS), 323
- Simon, T., 432
- SIMULA language, 1048
- coroutines, 222
  - inheritance hierarchies, 422, 429
  - logic programming, 556–557
  - object-oriented programming, 455
  - SMALLTALK and, 1049
- Simulation:
  - causal reasoning, 828–830
  - coroutines, 222
  - education applications of AI, 268
  - robotics programming, 940–941
  - temporal reasoning, 872
- Simulation and Evaluation of Chemical Synthesis (SECS), 94
- Single instruction stream, multiple data stream (SIMD) mode of execution, 678
- SIPE program, 754–755
- SIR system, 1048
- computational linguistics, 135
  - inheritance hierarchy, 1017–1018
  - pattern matching, 718
- Situational calculus:
  - temporal reasoning, 835–836, 871
- Situation description, military applications, 606–609
- Situation semantics, 1026–1027
- Skepticism, 282
- Sketchpad system and constraint satisfaction, 210
- Skewed symmetry and shape analysis, 1045–1046
- Skill acquisition tasks, intelligence, 438
- Skinner, B. F., 444
- Skolem functions, binary resolution, 895
- SL resolution:
  - belief revisions, inlist and outlist, 59
  - logic programming, 546
- Slagle, J., 4–6, 716–719, 762–766
- Slash operator, logic programming, 548
- SLIP utility, 1048–1049
- Slot-and-filler structures, artificial intelligence, 12–14
- Slots:
  - frame theory, 307–308
  - predicate calculus, 308–310
  - inheritance hierarchy, 422
  - constraints, 426
- SLUG (Symbolics LIST Users Group), 533

- Small-motion case analysis, 624
- Small processing elements (SPEs), 678
- Small, S. L., 204, 701-707
- SMALLTALK language, 1049
  - computer architecture, 215
  - control structures, 212-213
  - coroutines, object-oriented programming, 223
  - education applications, 268-269
  - frame theory, 309
  - human-computer interaction, 385
  - inheritance hierarchies, 422, 429
  - lambda calculus, 442
  - LISP machine, 529
  - logic programming, 556-557
  - object-oriented programming, 455
  - prostheses, 799
  - rule-based systems, 971
  - SIMULA and, 1048
- Smith, B. C., 1005-1010
- Smith, C. H., 409-416
- Smoothed local symmetries (SLS), shape analysis, 1044
- Smoothness assumption, optical flow, 686
- SMP system, 96
- SNePS system, 1049
  - frame language, 309
  - natural-language generation, 652
  - propositional networks, 1015-1016
  - semantic networks, 1013
  - assertional systems, 1019-1020
- SNIFFER system, 1049
- SNOBOL-4 language, 1049
- SNOBOL language:
  - GPSG parsers, 343
  - semantic networks, implementation, 1022
- SNPR program, 479
- SOAR project:
  - machine learning, 485
  - problem solving, 775
- Sobel operator, motion analysis, 623
- Social issues of AI, 502, 1049-1059
  - AI believers, 1052-1053
  - categorizing impact, 1050
  - disbelievers, 1053
  - economic issues, 1050, 1053-1055
  - future development, 1050-1051
  - interaction and office automation, 681
  - literature, 1059
  - political and control issues:
    - centralization/decentralization, 1055-1056
    - political and control issues, 1050, 1055-1057
  - psychological and issues, 1050, 1057-1058
  - robotics, 924
  - role constraints and natural language understanding, 675
  - scripts and ritualizations, 980-981
  - social critics, 1053
  - stereotypes, 838
  - technical spin-off issues, 1050, 1058-1059
- Social science, hermeneutics and, 369-370
- Societies for Artificial intelligence, 1060-1061
- Socratic method of teaching, computer-aided instruction, 154
- Software engineering, programming environments, 789
- Software protection, 460
- Solid (3D) models, 152
- Soloway, Elliot, 48
- Solution grammar, problem solving, 768-769
- Solution graph, problem reduction, 764
- Solution spaces, reasoning, 848
- Solution-space search, military applications, 606
- Sonar signals:
  - HASP system, 76
  - pattern recognition, 728
- SOPHIE system, 1061-1062
  - computational linguistics, 139-140
  - human-computer interaction, 385
  - intelligent computer-aided instruction, 154-155
  - natural-language understanding, 663, 667
  - semantic grammar, 350
- Soundness:
  - binary resolution, 895
  - completeness, 131
  - knowledge representation, 884
  - logic, 536
  - predicate logic, 543
  - theorem proving, 1116
- Sound synthesis, AI in music, 638
- SOURCE, INCORPORATIONRULES
  - knowledge system, 76
- Sowa, J., 1011-1023
- Space:
  - commonsense reasoning, 835
  - inference *vs.* description, 189-190
- Space exploration:
  - autonomous vehicles, 39
  - industrial robots, 949
  - mobile robots, 961
- SPADE programming tutor, 155
- Spaghetti stack:
  - control structure, 213
  - coroutine implementation, 220
  - InterLisp, 524
- SPAM system, image understanding, 398-400, 402
- Spatial knowledge:
  - image understanding, 391, 398-400
  - limits of artificial intelligence, 496-497
- Spatial layout, constraint satisfaction, 209
- Spatial reasoning, 863-869
  - commonsense reasoning, 835
  - quantities, 834
  - imagery, 867-868
  - ontology, 835
  - route finding and exploration, 866-867
- Spatial-planning dialogues, discourse understanding, 234-235
- Specialization, knowledge representation, 886
- Specific inference mechanisms, artificial intelligence, 15
- Specification in programming assistants, 785
- Specificity, decision theory, 229-230
- Spectral power distribution, 124-126
- Spectrograph, 744
- Specular reflection:
  - time-of-flight sensing, 932
  - triangulation, 932
- Speech acts, 1062-1063
  - belief systems, 68
  - Cohen and Levesque, 68
  - Cohen and Perrault, 68
- current problem areas, 1064-1065
- dialogue phenomena, 673
- direct mapping models, 1062-1063
- discourse understanding, 234, 241-243
- dynamic discourse model, 237
- goal determination inference, 675
- helpful response, 1063-1064
- indirect, 1062-1064
- machine translation, 565
- philosophy and, 743
- plan-based models, 1063-1064
- propositional content, 1062
- question answering, 816
- sincerity conditions, 1062
- Speech recognition, 1065-1070
  - artificial intelligence, limits of, 493-494
  - beam search, 56
  - consonant clusters, 1067-1068
  - human vocal tract apparatus, 1067
  - morphemes, 1068
  - robots, 720
    - industrial robots, 951
  - social issues of AI, 1056
  - understanding, 1076
  - Viterbi algorithm, 1160
  - wide-band spectrogram, 1066
- Speech synthesis, 1070-1076
  - constraints, 1070
    - human vocal apparatus, 1070
    - language structure, 1070
    - task, 1070
    - technology, 1070
  - evaluation, 1075
  - implementation, 1075-1076
  - letter-to-sound rules and lexical stress, 1073-1074
  - MITalk text-to-speech system, 1072
  - morphological analysis, 1072-1073
  - parsing, 1074
  - phonetic segments, 1075
  - prosodic framework, 1075
  - speech recognition, 1065-1070
  - stress modification and phonological adjustments, 1074
  - techniques, 1070-1071
    - parametric representation, 1071
    - synthesis-by-rule, 1071
    - waveform coding, 1070-1071
  - text analysis, 1072
  - text-to-speech conversion, 1072
- Speech understanding systems (SUSs), 1076-1082
  - acoustic-phonetic analysis, 1078-1079
  - ATNs, 327
  - case grammar, 333
  - current trends, 1081
  - domain knowledge, 252
  - echoic memory, 276-277
  - epistemology, 285
  - Harpy system, 57, 362, 384, 1068, 1079
  - HEARSAY-II system, 73-76, 213, 362, 384, 1069, 1079, 1081
  - HWIM system, 327, 384, 390, 1069, 1079, 1081
  - human-computer interaction, 384
  - ICAI, 158
  - image understanding and, 390
  - island-driven approach, 1081

- knowledge-source interpretation, 1079–1080
- language and, 503
- literature, 1081
- military applications of AI, 605
- pattern recognition, 727–728
- phonological analysis, 1079
- processing strategies, 1080–1081
- stress and intonation, 1078–1079
- system architecture, 1080
- theoretical background, 1077–1078
- Spelling errors:
  - natural-language understanding, 672
  - Viterbi algorithm, 1161–1162
- Spherical representation, path planning and obstacle avoidance, 711
- SpiceLisp, 526
- SPIE system, applications, 756
- Splines:
  - generalized cylinder representation, 321–322
  - space curve, 321–322
- Splits, inheritance hierarchy, 427–428
- SPOP arc, ATN grammar, 329
- Spreading activation:
  - cognitive modeling, 113–114
  - inheritance hierarchy, 429
  - knowledge representation, 886–887
  - semantic memory, 593
  - story analysis, 1091
- Spurious reification, 371–372
- Srihari, S., 1160–1162
- SRI International Development Corporation:
  - speech understanding, 1078
  - vision module, intensity image processing, 932–933
- SRL system:
  - demons, 429
  - inheritance hierarchies, 422
- SSS\* algorithm:
  - branch-and-bound search, 1003
  - branching factor, 82
  - pattern recognition, 726
- Stability:
  - descriptions, scale-space imaging, 978
  - game playing search reduction, 316
- Stable coalition, connectionism, 201–202
- Stage analysis, information processing, 436
- Standard LISP, 524
- Standard phrase structure derivation tree, 145
- Stanford Heuristic Programming Project, 607
- Stanford University (SU) Cart, 40
  - obstacle avoidance, 42
  - obstacle detection, 40
  - sensor mapping, 41
- Stanford University Robot Aid project, 799, 801–803
- STARAN system, associative memory, 17
- State-characterizing schemata, natural-language automatic programming, 31
- State space search:
  - artificial intelligence and, 10–12
  - beam search, 56–57
  - branch-and-bound search, 1003
  - horizon effect, 380
- path planning, decision point network, 709
- pattern recognition, 725–726
- processing:
  - bottom-up and top-down, 783
- State variables:
  - planning, domain assertions, 749–750
  - single-degree-of-freedom robot arm, 903
- Static evaluation function, game-playing, 316
- Static knowledge, path planning, 709–710
- Static-program analyzer programming tool, 793
- Static typing, programming environment, 791–792
- Statistical analysis, 419
  - default reasoning, 840–841
  - information retrieval, 419
- Statistical models:
  - texture analysis, 1103
  - background and foreground, 1104–1105
- STEAMER system:
  - human-computer interaction, 385–386
  - ICAI, 155–156, 158
  - military training, 611–612
  - question answering, 820
- Step transfer, legal analysis programs, 458–459
- Stepwise specialization, concept learning, 191
- Stereo vision, 1083–1089
  - computational stereo, 1083–1089
  - camera modeling, 1084–1085
  - distance determination, 1088–1089
  - feature acquisition, 1086
  - image acquisition, 1083–1084
  - image matching, 1086–1087
  - interpolation, 1089
  - search strategies, 1088
  - constraint satisfaction, 209–210
  - early vision models, 1139–1140
  - edge detection, 266
  - industrial robots, 951–952
  - navigation, 40
  - obstacle detection, 40
  - path planning and obstacle avoidance, 711
  - robotics direct range measurement, 930–931
  - template matching, 577
- Sternberg, R., 435
- Stick figures, art in AI, 9
- Stimulus-response theory of language acquisition, 444
- Stochastic processes:
  - inductive inference, 416
  - rule-based systems, 970
- Storage management, computer systems, 175
- Stored-program computers, Turing machines, 1125
- Story analysis, 1090–1098
  - inheritance hierarchy, 426
  - law applications, 461
  - memory organization packets, 592
  - world knowledge, goals and plans, 1093–1094
- Story molecules, 371
- Story understanding:
  - causal reasoning, 827–828
  - discourse understanding, 240–241
  - intelligent agents, 837
  - plan and goal structure, 837
  - SAM, 973
  - script applier, 987
- Straight-line correspondence, motion analysis, 623–624
- Strain-gauge wrist sensor, 937
- Strategic Computing Initiative (SCI), 604
  - political issues, 1056
- Strategy:
  - computer chess, 169
  - game playing, 313–314
  - minimax procedure, 615–616
  - multisensor integration, 636–637
- Stravinsky, Igor, 638
- Strawson, P. F., 542
- STREC system, 95
- Stress, AI and, 1054
- STRIPS MACROPS, 757
- STRIPS system, 9, 1099
  - belief revision, 58
  - heuristics learning, 475–476
  - means-ends analysis, 578, 581–582
  - path planning and obstacle avoidance, 708–710
  - pattern matching, 719
  - planning by, 757
  - temporal reasoning, 871
  - Hendrix's extension, 873
- STROBE system, domain knowledge, 253
- Strohlein, T., computer chess, 167
- Strong equivalence, parsing and grammar, 689
- Structural description, causal reasoning, 829
- Structural models, texture analysis, 1103
- Structuralism and hermeneutics, 363, 367
- Structured concepts and inheritance hierarchy, 422
- Structured light, robotics sensing, 931
- Structure-of-intellect model of intelligence, 433–434
- Structures and pattern matching, 717
- Structure of Scientific Revolutions, The*, 370
- Student model:
  - education applications of AI, 270
  - ICAI programs, 154
- STUDENT system, 1099
  - computational linguistics, 135, 140
  - pattern matching, 718
- Subjectivism and hermeneutics, 372
- Subsequent reference and natural-language generation, 645–646
- Subsumption:
  - goal, story analysis, 1094
  - theorem proving:
    - clause deletion, 1119
    - testing, 1117–1118
- Sufficiency criterion in cognitive science, 122
- Summarization in story analysis, 1093
- Summers, LISP synthesis, 26, 29
- Superquadrics, high level vision, 1149
- Surface boundary models, 152
- Surface cases, case grammar, 333
- Surface models, computer-aided design, 151–152
- Surface structure, natural-language generation, 651–652



- Susie Software, discourse understanding, 144
- Sussman, G., 502
  - CONNIVER, 205
  - HACKER program, 362
- Swartout, W., 329
- Switch rule, language learnability models, 446
- SWM system, belief reasoner, 60
- Syllabus, education applications of AI, 270
- Syllogistic reasoning, inheritance hierarchy, 422
- Symbolic computation:
  - chemistry, 96
  - computer architecture, 216–217
  - programming environment, 790
- Symbolic integration, AND/OR graphs, 7
- Symbolic logic:
  - computational complexity, 491
  - semantic networks, 1012
- Symbolic manipulation accelerator, connection machines, 200
- Symbolic relaxation algorithm, constraint satisfaction, 208
- Symbolic representation, cognitive science, 121
- Symbolic search, limits of, 497
- Symbolics 3600 LISP machine, 175, 525, 529
  - computer architecture, 216
  - computer system, 176–178
- Symbol manipulation procedures:
  - cognitive science, 121
  - Turing machines, 1124
- Symbols, LISP system, 511
- Symmetric axis transform (SAT) in shape analysis, 1044
- Symmetry detection, dot-pattern analysis, 255
- Synapses, 489–490
- SYNLMA systems, 94
- Syntactic analysis, 121
- Syntactic Structures*, 134, 360
- Syntactically driven parsing, 664–665
- Syntax:
  - belief systems, 63
  - case grammar, 338
  - computational linguistics, 134
  - deep structure, 230–231
  - hermeneutics, 366
  - human-computer interaction, 384
  - linguistic competence, 503–504
  - logic and, 536
  - machine translation, 566
  - modeling, 1108
  - natural-language analysis, 664–665
  - natural-language interface, 657
  - pattern recognition, 724–725
  - predicate logic, 539
    - deductive systems, 541
  - propositional attitudes, 837
  - propositional logic, 559–560
    - deductive systems, 561–562
  - semantics and, 351, 1024–1025
  - speech recognition, 1068
  - word-expert parsing, 702
- Synthesis, Greek concept, 775
- Synthesis-by-rule technique
  - speech synthesis, 1071
  - text-to-speech conversion, 1072
- SYSCONJ action, ATN grammar, 329–330
- System building, image understanding, 407–408
- Systemic grammar, 372
  - natural-language generation, 652–653
- Systems Control Technology, 607
- SYSTRAN system, 568–570
- Szolovitz, P., 440
- Tactical decision making, 605–606
- Tactile sensing:
  - industrial robots, 952
  - proximity, 804
  - robot sensors, 39, 493, 1035
    - artificial skin, 1036
  - contact sensors, 1036
  - force image, 1036
  - force sensing, 1035
- Tag information, LISP, 517
- Taie, M. R., 7, 233, 272, 362, 440, 571, 641, 687, 757, 796–797, 973, 1061–1062, 1166
- Tail recursion removal or merging, 513
  - LISP, 513
- Talking box, computers in education, 182–183
- TAMPR programming assistant, 788
- Tangled hierarchies, 422–423
  - frame theory, 307
- Target detection, template matching, 576
- Target language, programming assistants, 786
- Tarski decision problem, 492, 498
  - commonsense reasoning, 833
  - knowledge representation, 884
- Task analysis, human-computer interaction, 386
- Task decomposition, 247
  - functional *vs.* spatial, 247
  - hierarchical *vs.* heterarchical, 247
  - redundant *vs.* disjoint activities, 247
- Task encoding, robot-control systems:
  - feedback structure, 914
  - gradient dynamics, 914
  - methodology, 913–916
  - objective functions, 913–914
- Task knowledge, 294
- Task Oriented Dialogues, discourse understanding, 236
- Task structure, prostheses, 799
- TAUM system, 569
  - machine translation, 565
- Tautologies:
  - logic and, 536
  - propositional logic, 561
- Taxadvisor, 460
- TAXIS project, semantic networks, 1022
- TAXMAN I, 458–459
- TAXMAN II, 458–459
- Taxonomy, 481–483
  - inheritance hierarchy, 422
  - medical advice systems, 585
  - numerical, 482
  - semantic networks, 593–594
  - word-expert parsing, 703
- TDIDT (top-down induction of decision trees) systems, 470–471
- Teachable Language Comprehender (TLC), 443, 593
- Teaching, ICAI and, 158–159
- Teaching-by-doing, robotics programming, 938–940
- TEAM system:
  - natural-language interface, 657
  - natural language understanding, 668
- Technical reports, 530
- Tech program, computer chess, 164, 168
- Teddington Conference, machine translation, 566
- TEIRESIAS system, 586
  - description, 601–602
  - explanation facility, 600
  - knowledge engineering, 296–297
  - reasoning and, 601
  - rule-based systems, 971
- Telegram grammar, 651
- Telegraphic speech, 446
- Teleoperators, 1100–1101
  - robotics, 924
- Template matching, 576–577
  - hierarchical, 577
  - Hough transform, 383
  - pattern matching, 717
  - speech understanding, 1077–1078
  - word formation, 620
- Temporal reasoning, 870–874
  - artificial intelligence, 871–872
  - commonsense reasoning, quantities, 834
  - image understanding, 391, 399–400
  - interval logic, 874
  - logic, 618
  - philosophy and, 873
- Tenenbaum, J. M., 576–577
- Tennant, H., 272–273, 595–597
- Tenney, J., 638
- Term coordination, information retrieval, 420–421
- Term definition, predicate logic, 539, 542
- Term weighting, information retrieval, 420
- Terragator robot, 713
- Terseness principle, natural-language understanding, 673
- Terzopoulos, D., 1152–1159
- Tessellations in region-based models:
  - Delaunay, 1109–1110
  - Poisson line, 1109–1110
  - Voronoi, 1109–1110
- Testing, programming assistants, 785
- Test program sets (TPSs), equipment testing, 610
- Test-selection algorithms, medical advice systems, 588
- Tests, creativity and, 225
- Text analysis, 82–87
  - discourse understanding, 238–241
  - EPISTLE system, 287
  - limits of, 491
  - frame theory, 305
  - hermeneutics and, 364–365, 371
  - information retrieval, 420–421
  - office automation, 681–682
  - pattern matching, 716
  - speech synthesis, 1072
  - speech understanding, 1077
  - story analysis, 1090–1091

- structure analysis, 371
- Viterbi algorithm, 1161–1162
- Text generation:
  - computational linguistics, 144–145
  - grammaticality, 650
  - natural-language generation, 647–649
- Textons, 1110
- TEXT system and text generation, 145
- Texture analysis, 1101–1111
  - artificial intelligence, limits of, 493
  - feature extraction, 301
  - homogeneity, 1101
  - human perception models, 1110
  - inference of three-dimensional shape, 1110–1111
  - pixel-based models, 1103–1108
    - global models, 1104–1105
    - one-dimensional time-series models, 1103–1104
    - random field models, 1104–1108
  - region-based models, 1109–1110
  - shape analysis and, 1110–1111
  - shape-from-texture algorithms, 1111
- Theaetetus*, 63
- Thematic abstraction unit (TAU)
  - discourse understanding, 241
  - story analysis, 1097–1098
- Theorem proving, 1115–1122
  - AND/OR graphs, 7
  - applications, 1121–1122
  - automatic proving, 19–20
  - binary resolution, 893–894
  - circumscription, 102
  - commonsense reasoning, 834
  - completeness, 132
  - complete reduction sets, 1120–1121
    - confluence, 1120
    - critical pairs, 1121
    - termination, 1120
  - connection graphs, 1120
  - constraint satisfaction, 209
  - heuristics, 376–377
  - human-oriented systems, 1120
  - inference, 418
  - knowledge representation, 886
  - law applications, 457
  - logic, 536, 1115–1116
    - conjunctive normal form, 1116
    - notation, 1115–1116
    - one-literal rule, 1117
    - prenex normal form, 1116
    - Skolem normal form, 1116
    - special forms and concepts, 1116
    - terms and wffs, 1115
    - validity, 1116
  - logic programming, 545
  - machine translation, 565
  - matrix methods, 1120
  - MICRO-PLANNER, 604
  - nonmonotonic reasoning, 852
  - pattern matching, 719
  - problem reduction, 766
  - programming assistants, 785
  - reasoning, 822–823
    - set-of-support strategy, 824–825
    - subgoals, 826
  - resolution, 1118
    - linear resolution, 1118
    - smallest-first strategy, 1118
    - unit-preference strategy, 1118
  - self-reference, 1006
  - SNIFFER system, 1049
  - unification, 1119–1120
- Theory formation, problem solving, 767, 777–778
- Thinglab system, constraint satisfaction, 210
- Thinking Machines, Inc., 178
- Thompson, Ken, computer chess, 167
- Thrashing, constraint satisfaction, 207
- Three-dimensional image:
  - binocular procedure, 628–629
  - dot-pattern analysis, 254
  - feature correspondences, 621–622
  - industrial robots, 951–952
  - motion, generalization, 626–627
  - optical flow, 686
  - path planning and obstacle avoidance, 708–709
  - robotics, 940–941
  - shape analysis, 1046–1047
- 3-LISP, 213
  - self-reference, 1008
- Thresholding, edge detection, 1137
- Threshold logic units (TLUs), 204
- Throwness concept, 372
- Thurstone, L., 433
- Thyberg, Leslie, 184
- Tic-tac-toe:
  - natural-language generation, 646
  - search reduction, 315
- Time:
  - causal reasoning, 831
  - commonsense reasoning, 835–836
  - conceptual dependency rules, 195
  - see also* Temporal reasoning
- Time of flight sensing:
  - AM phase shift, 932
  - FM beat, 932
  - pulse time delay, 932
  - robotics direct range measurement, 930, 932
- Time-invariant systems, single-degree-of-freedom robot arm, 903
- Time Map Managing System, 873
- Time-series analysis:
  - texture analysis, 1103–1104
  - autoregressive model (AR), 1103
  - mixed model, 1103–1104
  - moving-average model (MA), 1103
- Time span, human-computer interaction, 385
- TINLAP meeting (1975), natural-language generation, 646–647
- Tip-of-the-tongue phenomenon, 599
- TMS system, *see* Truth maintenance systems
- Tolman, mental maps, 116
- Top-down analysis, cognitive science, 122
- Topological properties
  - feature extraction, 301
  - S connectivity, 301
  - shape representation, 835
- Top-to-bottom CF parsing, 689
- Torres y Quevedo, 159
- TORUS system, database interface, 138
- Touch and shape analysis, 1039–1047
- Touretzky, D. S., 422–430
  - redundant IS-A links, 425
- Tower of Hanoi Puzzle, 23–24
  - AND/OR graphs, 7
  - means-ends analysis, 579–580, 583
  - problem reduction techniques, 763
  - recursion, 876
- TQA system:
  - database interface, 138
  - parsing, 694
- Trace expert, natural-language automatic programming, 30–31
- Trainable desk calculator, 24
- Training tutorials, 681
- Trajectory planning, robot manipulations, 574–575
- Tranchell, L., 309
- TRANS system, primitives, 759
- Transformational analogy, 476–477
- Transformational generative grammar, 354
  - competence and performance, 503
- Transformational grammar, 133–134, 353–362
  - alternative theories, 359
  - ATNs, 325
  - base grammar, 355
  - computational models, 359–361
    - Marcus parser, 360–361
    - Petrack system, 360
  - sentence analysis, 360
  - deep structure, 230–231
    - parsing, 692
  - derivation process, 355–356
  - generative grammar, 354
  - language, 118, 354
    - acquisition, 444
  - machine translation, 566
  - natural-language processing, 661
  - natural-language understanding, 665–666
  - parsing, 142, 692–693
    - speed and efficiency, 694–695
  - pseudoinverse transforms, 692
  - structural descriptions, 345–347
  - surface structure parsing, 692
  - syntactic and semantic rules, 354
  - theories, 142
  - variations, 355–359
    - base grammar, 355–356
    - components, 356–357
    - constraints, 357–358
    - extended standard theory, 357
    - government-binding theory, 359
    - X-bar theory, 358–359
- Transformations:
  - automatic programming deductive mechanism, 20
  - transformational grammar, 354–355
- Transform goals, means-ends analysis, 579
- Transition rules, medical advice systems, 588
- Translation, machine, 133–134, 564–570
- Transposition tables:
  - computer chess methods, 164–165
  - software advances, 166–167
- Traveling-salesman problem, 777
- Tree grammar (TG), 339, 342
  - pattern recognition, 726
- Tree search:
  - mobile robots, 960

- problem reduction, 762
- and pruning, computer chess, 165–166
- semantic networks, 1017
- Waltz filtering, 1163–1164
- Tree-adjointing grammar (TAG), revival, 348–350
- Tree-structured graph, depth-first search, 1004–1005
- Treisman, 116
- Triangle tables, planning, 757
- Triangular conorms (T-conorms):
  - aggregation operations, 860–861
  - disjunction, 861
- Triangular norms (T-norms):
  - aggregation operators, 860–861
  - conjunction and propagation, 860–861
- Triangularity:
  - means-ends analysis, 583
  - mobile robots, 958
  - robotics direct range measurement, 930–932
- Trie, 86
- Trost, belief systems, 66
- Troubleshooting:
  - causal reasoning, 830
  - SOPHIE system, 1061–1062
- Truth conditions:
  - discourse understanding, 238–239
  - semantics, 1026
  - speech acts, 1062
- Truth-functional logic, 558–561
- Truth maintenance system (TMS):
  - backtracking, 48
  - belief revision, 58–59
  - changing belief sets, 60–61
  - premises and assumptions, 59
  - default reasoning, 842
  - uncertainty, 859
- Truth tables, propositional logic, 562–563
- Truth values, 537
  - belief systems, 67
  - propositional logic, 558
- Tsotsos, J. K., 389–407
  - control cycle, 395
- Tuceryan, M., 254–256
- Tuning parameters, stereo vision, 1087
- Turing, Alan, 9
  - AI literature, 530
- Turing machines, 1123–1125
  - analysis, 1123–1124
  - applications, 1125
  - ATN grammar, 329
  - parsing, 694
  - Church's thesis, 99–100
  - cognitive science, mechanism, learning and AI, 1125
  - computer chess methods, 159
  - definition, 1124–1125
  - feasible computations, 1125
  - Godel's theorem, 740
  - inductive inference, 410
  - LISP development, 520–521
  - Tower-of-Hanoi puzzle, 24–25
  - transformational grammar
    - parsing, 692–693
  - unrestricted rewriting systems, 689
- Turing test, 1126–1130
  - criticism, 1127–1128
  - history, 9
- imitation game, 1126–1127
- inductive inference, 1127
- legacy of, 1129–1130
- limits of artificial intelligence, 496
- narrowness, 1128–1129
- nature of, 1126–1127
- philosophy of mind, 736
  - antibehaviorist objection, 737
  - Block example, 737
  - objections to criteria, 736–737
- restricted, 1129–1130
- self-reference, 1006
- shallowness, 1128–1129
- support of, 1128
- thinking machines, 1126
- Turner, T., 306
- Tutoring, one-on-one, ICAI, 158
- Tutoring issue generator, 270–271
- Two-dimensional image:
  - optical flow, 684, 686
  - path planning and obstacle avoidance, 708
  - shape analysis, 1044–1046
- 2 1/2 D sketch, 1142, 1153
- Type-identity theorists, 737–738
- Type 0 grammars, 766
- Typicality judgements:
  - default reasoning, 840–841
  - nonmonotonic reasoning, 849
- UCI LISP, 524
- UCLA grammar, 142
- UC system, office automation, 140
- Ultracentrifugation, chemical instrumentation, 96
- Uncertainty:
  - aggregation and, 854
  - conjunctions and disjunctions, 860
  - Bayesian decision methods, 51–52, 855
  - modification, 855–856
  - belief theory, 857–858
  - blackboard architecture, 74–75
  - calculi, 861
  - certainty factor, 856–857
  - early vision, 1139
  - endorsements theory, 859
  - evidence space, 857–858
  - evidential reasoning, 858
  - granularity, 860
  - image understanding system, 399–400
  - inductive inference, 415
  - necessity and possibility theory, 858
  - numerical approaches, 854
  - plausible reasoning:
    - expert systems, 854
    - sources, 854
  - programming environment, 790, 796
  - reasoning and, 854–855, 859–860
  - representations compared, 859–860
  - sensor, 41
  - single-degree-of-freedom robot arm, 903
  - symbolic representations, 854–855
- Undecidability and predicate logic, 543
- Undersea mobile robots, 957, 961
- Understanding:
  - discourse, 233–244
  - frame theory, 304
  - Gadamer's philosophy, 365–366
  - Heidegger's philosophy, 364–365
- hermeneutics and, 363–364, 372–373
- image, 389–407
- machines, 681–682
- natural-language, 660–675
- Unification:
  - algorithm, 413
  - ATNs, 329
  - automatic programming deductive mechanism, 20
  - connection machines, 200
  - inductive inference, 413
  - knowledge representation, 886
  - logic programming, 546
  - pattern matching, 717, 719
  - problem reduction, 764
  - processing, bottom-up and top-down, 784
  - theorem proving, 1119–1120
  - word formation, 620
- Unification-based grammar, 692
- Uniform object-oriented languages, 455
- Unilateral paralysis, aids for, 803
- UNIMEM system, 483
- UNITS system, 95
- Unit-value principle, connectionism, 201–203
- Universal grammar, linguistic competence and performance, 504
- Universal pragmatics and hermeneutics, 366
- Universal quantifier, predicate logic, 539
- UNIX system:
  - Drilling Advisor system, 291
  - office automation, 140
- Unrestricted rewriting systems, 689
  - ATN parsing, 694
- Unsolvability concept, 491–492
- Unsupervised learning, 187
- Universal grammar, 354
- UPL system, 475
- Use-machine-environment triad for prostheses, 798
- User interface:
  - computer systems, 175
  - human-computer interaction, 384–385, 655
  - office automation, 680–682
- User interface management system (UIMS), 387
- User modeling:
  - belief systems, 63, 70
  - information retrieval, 421
- USL system database interface, 138
- Utility values
  - Bayesian decision methods, 55
  - decision theory, 229
- VAL language:
  - industrial robots, 947
  - prostheses, 799
- Validation:
  - learning concepts, 191
  - legal analysis programs, 459
  - medical advice systems, 587
- Value cells in LISP, 511
- Values:
  - Bayesian decision methods, 55–56
  - causal reasoning, 829–830
- Van Gulick, R., 736–743

- Van Riemsdijk, J., 359
- Variable-structure systems theory, robot control, 917–918
- Variables:
- expectation-driven parsing, 698–699
  - LISP, 511
  - LISP 1.5, 521–522
  - pattern matching, 716–717
  - predicate logic, 538
  - qualitative physics, 810
- VAX computers, 10
- case frame ellipsis resolution, 673
  - connection machines, 178–179
  - knowledge engineering, 288, 297
  - LISP development, 525–526
  - OPS-5, 684
- Vehicles, autonomous, 39–44
- Velocity field, optical flow, 684
- Venken, R., 46–47, 219–223
- Verb-centered relational graphs, 1013–1014
- Vercoe, B., 638
- Vere, S., 748–757
- Vernon, D., 434
- Veroff, R., 892–901
- Version-space search method, 469–470
- generalized plans, 478–479
  - heuristics learning, 475
- Very Large Scale Integration (VLSI) technology, 175
- Veterans Administration, prostheses development, 800
- Vidicon color imaging, 125
- Vilain's system of time maintenance, 873
- VIPS system, office automation, 140
- VIR arcs, ATN grammar, 328, 330
- Virtual machines:
- control structures, 211
  - generalization, 213–214
  - frame theory hierarchies, 307–308
- Viscous damping, robot-control systems, 912
- Visicalc system, constraint satisfaction, 210
- Vision*, 396
- Vision:
- anthropomorphic robots, 945
  - artificial intelligence, limits of, 493
  - beam search, 56
  - belief revision, 61
  - cognitive science, 121–122
  - connectionism, 200
  - connection machines, 200
  - constraint satisfaction, 209
  - dot-pattern analysis, 253
  - early, 1131–1149
    - abstraction and general-purpose models, 1138–1139
    - algorithms, 1145
    - controlled hallucination, 1137
    - cooperative process and energy minimization, 1139–1140
    - depth map, 1152
    - discontinuity and edge detection, 1134–1135
    - edge detection and scale, 1135
    - elusive edge operator, 1135–1136
    - figure-ground and structure grouping, 1147–1148
    - function, 1145–1146
    - general-purpose models, 1144–1145
    - inference mechanisms, 1145
    - intensity discontinuities, 1145
    - interacting modules from surface interpolation, 1142
    - intrinsic images, 1142
    - inverse optics, 1141
    - knowledge-based edge detection, 1137–1138
    - lateral inhibition, 1133–1134
    - naive physiology, 1136–1137
    - neural modeling, 1139
    - organization and complexity, 1138–1139
    - parallelism, 1139
    - primal sketches, 1142
    - primate system organization, 1134–1135, 1143–1144
    - qualitative *vs.* quantitative knowledge, 1144
    - rigidity, 1138
    - Roberts's system, 1132–1133
    - segmentation to description, 1139
    - segmentation to surfaces, 1141–1142
    - surfaces, 1147
    - symbolic side, 1142–1143
    - tasks, tools, techniques, 1141
    - texture, 1147
    - uncertainty, 1139
  - epistemology, 285
  - high level, 1149
    - commonsense reasoning, 833
  - image understanding, 389
  - industrial robots, 948, 951
  - later, 1131
  - limits of artificial intelligence, 493
  - model-based, 492
    - shape analysis, 1046–1047
  - multisensor integration, 632
  - problem solving, 776
  - proximity sensing, 804
  - robot sensors, 39, 1033–1034
    - motion vision, 1034
    - static vision, 1033–1034
  - scripts, 984
  - shape analysis, 1039–1040
    - boundary descriptions, 864
  - social issues of AI, 1051
  - spatial, 835, 1143–1144
    - parallelism, 1145
  - template matching, 576
  - texture analysis, 1101
  - three-dimensional, social issues of AI, 1051
    - see also* Perception
- VISIONS system:
- image understanding, 397, 399, 402
  - representational formalisms, 398–399
- Visual depth map, 1152–1159
- depth acquisition, 1153
  - depth representation, 1153–1154
  - reconstruction, 1154–1158
    - algorithms, 1157–1158
    - discontinuities, 1155
    - finite-element method, 1158
    - generalized spline, 1155
    - interpolation, 1154–1155
    - membrane spline, 1155
    - multigrid relaxation methods, 1158
    - multiprocess integration, 1154
    - multiscale fusion, 1156
    - penalty functional, 1157
    - piecewise-constant image restoration, 1156
    - stabilizing functional, 1157
    - thin-plate surface under tension, 1155–1156
- Visual impairment, prostheses for, 800–801
- Viterbi algorithm, 1160–1162
- heuristic modifications, 1161
  - probabilistic criteria, 1161
  - spelling correction and text recognition, 1161–1162
  - stereo vision, 1088
- Vividness and episodic memory, 278
- V language, automatic programming construction, 33
- VLISP, word-expert parsing, 703
- VLSI computers:
- associative memory, 17–18
  - beam search, 57
  - computer aided design, 153
  - connection machine, 177–178
  - expert systems, 288
  - industrial robots, 952
  - limits of artificial intelligence, 494
  - robotics sensing, 935–936
  - shape analysis, 1044
  - tactile sensing, 1036
- VM system:
- medical advice systems, 587
  - research issues, 588
- Vocabulary:
- intelligence and, 432
  - law applications, 461
- Voghera, N., 132
- Voice mail, 682
- Voice synthesizers, computers in education, 182–183
- Volumetric description, 863–864
- robotics sensing, 935
- von Foerster, H., 225–227
- von Helmholtz, Hermann, 389
- early vision, 1131–1132
- von Neumann architecture, 174
- von Neumann bottleneck, 174
- cell-space model, self-replication, 1010–1011
  - Church's thesis, 99–100
  - computer systems, 174
  - connectionism, 200
  - control structures, 211–212
    - programming language design, 212
  - rule-based systems, 965
  - Symbolics 3600, 177
  - Turing machine, 1125
- Von Restorff effect, episodic memory, 278
- Voronoi tessellation, dot-pattern analysis, 255
- Voting, Hough transform, 383
- Votrax speech-production system, 645–646
- Voxels, volumetric description, 863–864
- WALK system, 307
- Wallace, V., 387
- Wallas, G., 224
- Wall following, path planning and obstacle avoidance, 714
- Walter, D., 881
- Waltz filtering, 1162–1165

- assessment of applications, 1165
- constraint satisfaction, 208
- early vision, 1140
  - labeling line drawings, 1162-1163
- Waltz, D. L., 204, 1162-1165
- Wand, M., 441-442
- Wandering standpoint algorithm, mobile robots, 960
- Wang's algorithm, propositional logic, 562
- WARP machine, computer architecture, 216
- Warren instruction set, 216
- Waterman, D. A., 459-460
- Water chemistry, 96
- Watkins, M., 275-279
- Waveform coding, speech synthesis, 1070-1071
- WAVES system, 297
- Weaver, Warren, 133, 566
- Web and graph grammars, pattern recognition, 726
- Webber, B., 814-821
- Wechsler Adult Intelligence Scale, 432
- Weighting factors, language acquisition, 449-450
- Weizbaum, 1053
  - social issues of AI, 1056
- Well-formed propositions (WFPs), 561
- Well-formed formula, predicate logic, 539, 542
- Wertheimer, 115-116
- Wesson's air-traffic-control planner, 756
- Westinghouse Science Talent Search, 225
- WEST system, 155
  - education applications of AI, 269-270
  - human-computer interaction, 385
  - tutoring issue generator, 270-271
- Wexler, K., 445
- What-if capability, natural image language interface, 659
- Wheelchairs, 797-798
- Whiskers, path planning and obstacle avoidance, 714
- WHISPER program, spatial reasoning, 867-869
- White, John I., 523
- Whole-task analysis, information processing, 436
- WHY program, 155
- WICAT School, 181
- Wide-spectrum language, automatic programming, 33
- Wiener, Norbert, 530
  - cybernetics, 225
- Wilensky:
  - frame manipulation primitives, 305
  - text comprehension principles, 305
- Wilks, Y., 564-570, 759-761
  - machine translation system, 570
- Williams, E., 359
- Winner-take-all (WTA) net, 202
- Winograd, Terry, 10
  - ATNs, 324
  - hermeneutics, 372
- knowledge representation language (KRL), 371-373
- MICRO-PLANNER, 603-604
  - music in AI, 641
- Winston, P.:
  - blocks world, 190
  - frame theory, 302
  - learning and reason by analogy, 373
- Wire-frame models, 152, 890-892
  - optical flow, 685
  - robotics programming, 940
- Wiswesser line notation, 93
- Witkin, A., 973-979
- WOK system, 592
- Woods, W. A., 323-332, 1029-1030
  - augmented transition network (ATN), 136, 666
  - discourse structure, 239
  - LUNAR, 137, 564
  - natural language understanding, 655
  - procedural semantics, 136, 1027
  - semantic networks, 883, 1012
  - speech and image understanding, 390
- Woodham, R. J., 1039-1047
- Word disambiguation, frame theory, 304-305
- Word-expert parsing, 701-707
  - abbreviated execution trace, 706
  - content and function words, 703
  - context-probing, 703-704
  - control mechanisms: expert suspension and resumption, 705
  - example analysis, 705
  - example sentence, 706
  - expectation-driven parsing, 699
  - formalizing, 703-704
  - message and memory objects, 705
  - model organization, 705
  - principles, 702
  - structure, 704
  - working memory after analysis, 706-707
- Word factory, computers in education, 182-183
- Word formation:
  - morphology, 619-620
  - two-level model, 620
- Word perception theory, 87
- Word processing, computers in education, 182
- Word sense, 593
- Word-sense discrimination, word-expert parsing, 702, 705
- Word shape theory, 87
- Word-for-word processing, 133
- Working at home, office automation, 683
- Working memory:
  - DADO computer, 228
  - expectation-driven parsing, 700
- Work station implementation, natural language interface, 659
- Word formalism, distributed problem solving, 249
- World knowledge, 294
- acquisition, 1096
- connectionism, 200
- story analysis:
  - goals and plans, 1093-1094
  - scripts, 1092-1093
- Wos, L., 892-901
- Wright, J. M., 309
- Writer's Workbench, 140
- WUMPUS system:
  - human-computer interaction, 385
  - ICAI, 155
- X-bar theory:
  - grammar learning, 479-480
  - transformational grammar, 358-359
  - chemical structure elucidation, 95
- XCALIBUR project, 673-674
  - semantic grammar, 350
- XCON system, 10, 1166
  - certainty factor, 14
  - knowledge engineering, 297
  - OPS-5, 684
  - rule-based systems, 971-972
- XPLAIN system:
  - explanation, 299
  - medical advice system, 586
- XPROBE system, prostheses, 799
- X-rays, 727
- XSEL system, 673
  - knowledge engineering, 297
- X-Y-Z set of colors, 125
- Yale Artificial Intelligence Project, 591-592
- Yngve, natural-language generation, 650
- Yuhan, A. Hanyong, 205, 312, 323, 362, 441, 488, 564, 973, 1048-1049, 1099
- ZBIE system, language acquisition, 145-146
- Zero crossings:
  - edge detection, 1136
  - feature acquisition, 1086
  - intensity change detection, 257, 261
  - scale-space imaging, 974-976
  - two-dimensional, 979
  - stereo vision, 1087
- Zero order logic, 538
- Zero-sum two-person games, minimax procedure, 615-617
- ZetaLisp, 525-526
  - history, 526
  - Symbolics 3600 computer system, 176-177
- ZMACS editor, 529
- ZMOB machine, symbolic computation, 216
- Zucker, S. W., 1131-1149
- Zugzwang positions, computer chess, 162-163
- Zuse, 159
- Zwicky, A., 360